Directional Control of Generating Brownian Path under Quasi Monte Carlo

by

Kai Liu

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Actuarial Science

Waterloo, Ontario, Canada, 2012

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Quasi-Monte Carlo (QMC) methods are playing an increasingly important role in computational finance. This is attributed to the increased complexity of the derivative securities and the sophistication of the financial models. Simple closed-form solutions for the finance applications typically do not exist and hence numerical methods need to be used to approximate their solutions. QMC method has been proposed as an alternative method to Monte Carlo (MC) method to accomplish this objective. Unlike MC methods, the efficiency of QMC-based methods is highly dependent on the dimensionality of the problems. In particular, numerous researches have documented, under the Black-Scholes models, the critical role of the generating matrix for simulating the Brownian paths. Numerical results support the notion that generating matrix that reduces the effective dimension of the underlying problems is able to increase the efficiency of QMC. Consequently, dimension reduction methods such as principal component analysis, Brownian bridge, Linear Transformation and Orthogonal Transformation have been proposed to further enhance QMC. Motivated by these results, we first propose a new measure to quantify the effective dimension. We then propose a new dimension reduction method which we refer as the directional method (DC). The proposed DC method has the advantage that it depends explicitly on the given function of interest. Furthermore, by assigning appropriately the direction of importance of the given function, the proposed method optimally determines the generating matrix used to simulate the Brownian paths. Because of the flexibility of our proposed method, it can be shown that many of the existing dimension reduction methods are special cases of our proposed DC methods. Finally, many numerical examples are provided to support the competitive efficiency of the proposed method.

Key words:

QMC, Low Discrepancy Sequence, Effective Dimension, Dimension Reduction, PCA, BB, LT, OT, FOT, DC

Acknowledgements

I would like to express my sincere thanks and gratitude to my supervisor, Prof. Kan Seng Tan, for preparing this thesis. He is truly exceptional and I am really thankful that I had the opportunity to learn from him.

I would also like to thank my thesis committee members Prof. Kolkiewicz and Prof. Lemieux for taking their time and effort to read my thesis and provide insightful comments despite their extremely busy schedules.

Dedication

This is dedicated to the ones I love, especially my parents. I could not have completed it without their unconditional support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The central theme of this thesis is to propose a QMC-based simulation method to increase the efficiency of the QMC in estimating high-dimensional integrals associated with pricing the derivative securities. Because of the complexity of derivative securities and the sophistication of financial models, the integrals associated with finance applications typically cannot be evaluated analytically. Consequently Monte Carlo (MC) method, which is first introduced in quantitative finance by Boyle (1977), becomes a popular numerical method. However, MC method attains a convergence rate of $O(1/\sqrt{N})$ where N is the number of simulation trials . MC method is often criticized to be a slow method despite the convergence rate is independent of the dimension. In mid 1990s, several reports have surfaced advocating the use of Quasi-Monte Carlo (QMC) methods, as opposed to the classical MC methods. QMC offers a convergence rate of $O\left(N^{-1}(\log N)^d\right)$ in dimension d. This rate is asymptotically more efficient than the MC. The results in Joy et al. (1996) and Paskov and Traub (1995) showed that QMC yields a much higher accuracy than the Monte Carlo (MC) method, even for several hundred dimensions. As a result of these findings and the theoretically more efficiency (than MC), there is a surge of interest among among financial engineers and academicians in using QMC methods to computational finance. One key area of research focus is to provide a better understanding on why QMC can be effec-

tive in finance applications. Another area of research focus is to seek better QMC-based algorithms for evaluating high-dimensional integrals.

We now provide a brief overview connecting MC and QMC methods to pricing European derivative securities. For a detailed exposition of these topics, see Glasserman (2004) and Lemieux (2009). For simplicity, we assume the the dynamics of the asset price is governed by the Black-Scholes (BS) model so that the risk-neutral process of the underlying asset S_t at time t, is given by

$$dS_t = rS_t\,dt + \sigma S_t\,dB_t, \tag{1.1}$$

where r is the risk-free interest rate, σ is the volatility and B_t is the standard Brownian motion.

By $h(\mathbf{S}) = h(S_1, \ldots, S_d)$ we denote as the payoff function of a particular derivative security at maturity T years. Note that the payoff function depends on the asset prices $S_j := S_{t_j}$ at equally spaced times $t_j = j\Delta t$ for $j = 1, \ldots, d$ and $\Delta t = T/d$. According to the option pricing theory, the value of the financial derivative at $t = 0$ is

$$\mathbb{E}\,[e^{-rT} h(\mathbf{S})], \tag{1.2}$$

where $\mathbb{E}\,[\cdot]$ is the expectation under the risk-neutral measure. For example, the price of a European arithmetic Asian option is $\mathbb{E}\,[e^{-rT}\,\max(S_A - K, 0)]$, where S_A is the arithmetic average of the underlying asset prices at times $t_1, \ldots, t_d$ and K is the strike price.

It is easy to verify that the payoff function $h(\mathbf{S})$ can be re-expressed as

$$h(\mathbf{S}) = h(\exp(\mu_1 + \sigma x_1), \ldots, \exp(\mu_d + \sigma x_d)) := g(\boldsymbol{x}), \tag{1.3}$$

where $\mu_j = \log S_0 + (r - \sigma^2/2)\,t_j$, $\boldsymbol{x} := (x_1, \ldots, x_d)^T \sim N_d(\mathbf{0}, \boldsymbol{\Sigma})$; i.e, $\boldsymbol{x}$ is normally distributed with mean $\mathbf{0}$ and covariance matrix $\boldsymbol{\Sigma}$ with its entry given by

$$\boldsymbol{\Sigma}_{ij} = \min(t_i, t_j) = \Delta t\,\min(i, j). \tag{1.4}$$

Note that in (1.3) we have redefined $h(\mathbf{S})$ as $g(\boldsymbol{x})$ to emphasize the explicit role of $\boldsymbol{x}$. Consequently, the time-0 value of the financial derivative can be expressed as a Gaussian

integral:

$$V(g) := \mathbb{E}\left[g(\boldsymbol{x})\right] = \frac{e^{-rT}}{(2\pi)^{d/2}\sqrt{\det \boldsymbol{\Sigma}}} \int_{\mathbb{R}^d} g(\boldsymbol{x}) \exp\left(-\frac{1}{2}\,\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x}\right) d\boldsymbol{x}. \tag{1.5}$$

From the point of view of integration, by setting $\boldsymbol{x} = \boldsymbol{A}\,\boldsymbol{z}$, where $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{\Sigma}$, $\boldsymbol{z} = (z_1, \ldots, z_d)^T \sim N_d(\boldsymbol{0}, I_d)$ and I_d is the $d \times d$ identity matrix, and then imposing the transformation $\boldsymbol{z} = \Phi^{-1}(\boldsymbol{u}) = (\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_d))^T$, the componentwise inverse of the standard normal cumulative distribution function, the Gaussian integral (1.5) is transformed to

$$V(g) = \frac{e^{-rT}}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} G(\boldsymbol{A}\boldsymbol{z}) \exp\left(-\frac{1}{2}\,\boldsymbol{z}^T\boldsymbol{z}\right) d\boldsymbol{z} = e^{-rT} \int_{(0,1)^d} G(\boldsymbol{A}\,\Phi^{-1}(\boldsymbol{u}))\, d\boldsymbol{u}. \tag{1.6}$$

The change of variables $\boldsymbol{x} = \boldsymbol{A}\,\boldsymbol{z}$ is equivalent to the generation of the Brownian motions

$$(B_1, \ldots, B_d)^T = \boldsymbol{A}\,(z_1, \ldots, z_d)^T, \tag{1.7}$$

where B_j is the Brownian motion at time t_j. Hence we refer to $\boldsymbol{A}$ as the *generating matrix* of the Brownian motion. A key insight to the above transformation is that $\boldsymbol{A}$ can be arbitrary as long as it satisfies $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{\Sigma}$. Consequently, different specification of $\boldsymbol{A}$ yields different methods of generating the Brownian motions. These methods are commonly known as the path generating methods (PGMs) since they relate to the simulation of Brownian paths.

By defining $\boldsymbol{z}_k = \Phi^{-1}(\mathbf{u}_k)$ and $\mathcal{P} := \{\mathbf{u}_i, i = 1, \ldots, N\}$ is a low discrepancy point set over the unit cube $(0,1)^d$, the QMC estimate of (1.6) is given by

$$Q(g, \boldsymbol{A}, \mathcal{P}) = \frac{e^{-rT}}{N} \sum_{k=1}^{N} G(\boldsymbol{A}\,\boldsymbol{z}_k) = \frac{e^{-rT}}{N} \sum_{k=1}^{N} G(\boldsymbol{A}\,\Phi^{-1}(\mathbf{u}_k)),$$

Clearly, the accuracy of the above estimate depends on the point set $\mathcal{P}$, the payoff function g and the generating matrix $\boldsymbol{A}$. While it is known that the MC algorithms based on different PGMs, i.e. different $\boldsymbol{A}$, are equivalent since the mean square error of MC is determined by the variance of the integrand, which is unchanged, different PGMs have significant impact on QMC. This phenomenon arises as the resulting function $G(\boldsymbol{A}\,\Phi^{-1}(\boldsymbol{u}))$ may have different dimension structure and may also induce different smoothness property depending on the

chosen $\boldsymbol{A}$. In particular, it is widely believed that techniques that reduce the effective dimension of the function G increases the efficiency of QMC. As such, many dimension reduction techniques such as the Brownian bridge (BB) (Moskowitz and Caflisch 1996), the principal component analysis (PCA) (Acworth et al 1998), the linear transformation (LT) (Imai and Tan 2006), orthogonal transformation on discontinuous function (OT) (Wang and Tan 2012) and fast orthogonal transformation (FOT) method (Leobacher 2010), have been proposed to increase the efficiency of QMC.

In high dimensions, the powers of $\log(N)$ in the convergence rate of QMC are not negligible for practical sample sizes; i.e. the theoretical higher asymptotic convergence rate of QMC is not achievable for practical applications in high dimensions. In non-finance applications, Bratley et al. (1992) indicated that QMC offers no practical advantage over MC, even for problems with low dimensions. Several explanations have been offered to reconcile the conflicting results. Based on the notion of tractability and strong tractability, Sloan and Wozniakowski (1998, 2001) showed that there exists a QMC algorithm for which the curse of dimensionality is not present in some weighted function classes. Papageorgiou (2001) and Owen (2001) demonstrate the superiority of QMC in some isotropic integrals. Another important reason for the success of QMC in high dimensions is the distinction between nominal dimension and effective dimension. Using the "analysis of variance" (ANOVA) decomposition of a function, Caflisch et al. (1997) defined two notions of the effective dimension: truncation dimension and superposition dimension. Essentially, the truncation dimension indicates the number of important variables which predominantly capture the evaluation function f, and the superposition dimension measures to what extent the low-order ANOVA terms dominate the function. However, in general, it is hard and computationally inefficient to compute the effective dimension for an arbitrary function, which we will discuss in Chapter 3, so seeking for an efficient substitute is important.

Papageorgiou (2002) pointed out that any decomposition $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{\Sigma}$ provides a construction for a discrete approximation of a Brownian path via $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{Z}$, where $\boldsymbol{Z}$ is a standard normal vector. In this context, the forward construction corresponds to the Cholesky decomposition of $\boldsymbol{\Sigma}$. However, Sloan and Wang (2010) show an "equivalence principle",

which roughly states that every decomposition is equally bad and good for QMC, depending on the function that one wants to integrate. In other words, every decomposition that is good for one payoff function is bad for another.

This thesis aims to provide a newly designed directional control (DC) method to enhance the evaluation of high-dimensional financial problems for choosing good constructions with different payoff functions. It also discusses the relationship between DC and other methods.

The layout of thesis is as follows: Chapter 2 reviews some of the common low discrepancy sequences. Chapter 3 discusses the dimension-wise decomposition, and in particularly the notion of effective dimensions. Chapter 4 first reviews some popular PGMs and then describes our proposed DC method. We will also demonstrate that our proposed DC method can recover all of existing PGMs. Chapter 5 gives some numerical results to assess the relative efficiency of the PGMs. Last chapter concludes the thesis as well as states some directions for future research.

Chapter 2

Low Discrepancy Sequence

The MC method to evaluating multidimensional integrals involves computing the integrand at a sequence of N random sequences and computing the average of the integrand values. The MC method has an integration error of the order of $O\left(N^{-\frac{1}{2}}\right)$, which is independent of the number of dimension d. However, the error bound implies that to obtain one additional decimal of accuracy, it is necessary to increase the number of points from MC sequences by a hundred times. In the last few decades, the number theory research has led to the development of a more efficient simulation method, which is called quasi-Monte Carlo (QMC) method. It uses the basic principle of the MC method to evaluate a multi-dimensional integral by the average of the values of the integrand evaluated at discrete points. However, rather than using sequences that are randomly generated, QMC method relies on deterministically generated low discrepancy sequences. These sequences are designed to achieve a more even distribution of points in the integration space than the random and pseudo-random sequences.

In particular, the theoretical upper bound for the integration error in the QMC method is of the order $O\left(N^{-1}(\log(N))^d\right)$, where N is the number of QMC sequences. However, despite the fast convergence of QMC, the QMC method has two major limitations:

1. The deterministic nature of the quasi-random sequences makes it difficult to estimate

the error in the QMC simulation procedure.

2. Quasi-random sequences exhibit poor properties in higher dimensions.

Due to the above limitations of QMC, a growing field of research in QMC has resulted in the development of efficient randomization strategies to estimate the error in the integral evaluation and scrambling techniques to "destroy" somewhat the correlations in higher dimensions.

In this chapter, our purpose is to review some of the properties of QMC sequences, and discuss some improved strategies for constructing better sequences. Section 2.1 discusses the QMC sequences, Section 2.2 describes some scrambling techniques, and Section 2.3 focuses on the randomization strategies. The last section will discuss the error bound of QMC sequences.

2.1 QMC Sequences

The fundamental idea of using QMC sequences is to evaluate a multi-dimensional integral by computing the average value of the integrand over a sequence of low-discrepancy points. More precisely, the definition of star discrepancy is defined as follows:

$$D_N^* = \sup_{J\in[0,1)^d} D_N(J) = \sup_{J\in[0,1)^d} \left| \frac{number\ of\ points\ in\ J}{N} - v(J) \right|, \tag{2.1}$$

where $J = z_1, ..., z_N$ is a set of points, and v is the volume of the unit cube, which is also defined as the d-dimensional Lebesgue measure. The (star) discrepancy can be thought of as the greatest absolute difference between the continuous uniform probability and the discrete uniform probability taken over all possible sub-cubes of $[0, 1)^d$ containing the origin.

In one-dimensional case: If $0 \le x_1 \le ... \le x_N \le 1$, then

$$D_N^*(x_1, ..., x_N) = \frac{1}{2N} + \max_{1\le n\le N} \left| x_n - \frac{2n-1}{2N} \right| \le \frac{1}{2N}. \tag{2.2}$$

This implies that we always have $D_N^* \leq \frac{1}{2N}$ and with equality if $x_n = \frac{2n-1}{2N}$. i.e. midpoint rule.

For an infinite one-dimensional sequence of points, $D_N^* \leq c\frac{\log N}{N}$ for some constant c which does not depend on N.

For a d-dimensional point set or sequence, we have the following Roth (1954) lower bound:

$$D_N^* \leq \hat{B}_d \frac{\log^{(d-1)/2} N}{N} \textit{ for a finite point set} \tag{2.3}$$

$$\textit{and } D_N^* \leq \tilde{B}_d \frac{\log^{(d)/2} N}{N} \textit{ for a finite point set,} \tag{2.4}$$

where $\hat{B}_d, \tilde{B}_d > 0$ depends only on d.

When a point set satisfies the conjectured lower bound, then we denote such point set as the low discrepancy point set. i.e. Low discrepancy point set has the following lower bound: $D_N^* = O\left(\frac{\log^{d-1} N}{N}\right)$. Similarly, for an infinite sequence, low discrepancy point set has this lower bound: $D_N^* = O\left(\frac{\log^{d} N}{N}\right)$. In particular, when $N \to \infty$, $\lim_{N\to\infty} D_N^* = 0$.

Many of the low-discrepancy sequences in use today are linked to the van der Corput sequence, which was originally introduced for dimension $d = 1$ and base $b = 2$ (van der Corput 1935). An alternative approach to generate low-discrepancy sequences is to start with points placed into certain equally sized volumes of the unit cube. These fixed length sequences are referred to as (t, m, d)-nets, and related sequences of indefinite lengths are called (t, d)-sequences. Sobol (1967) suggested a multidimensional (t, d)-sequence using base 2, which was further developed by Faure (1982) who suggested alternate multidimensional $(0, d)$-sequences with base $b \leq d$.

Figure 2.1 represents dimension 1 and dimension 2 of the first 128 points of Sobol sequence in 128 dimensions with Matlab function sobolset(128). It should be clear that these points are well dispersed over the unit square, consistent with the greater uniformity property of the Sobol sequence.

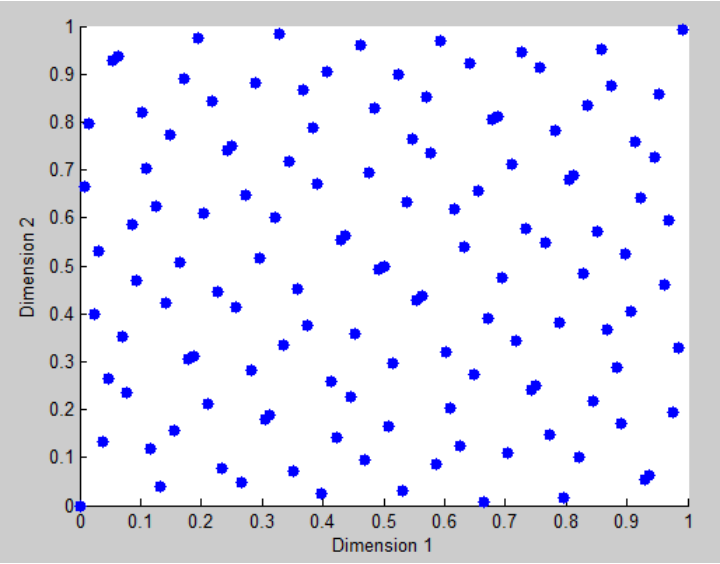

Figure 2.1: Dimension 1 vs Dimension 2 for $N = 128$

Figure 2.1 represents dimension 1 and dimension 50 of the first 128 points of Sobol sequence in 128 dimensions with Matlab function sobolset(128). In this case, these points are no longer well-dispersed over the unit square. This phenomenon is attributed to the high dimensionality of the sequence (i.e. 128th dimension versus second dimension in the previous graph). Hence using these sequences for evaluating high-dimensional integrals can be problematic. This phenomenon was pointed out Morokoff and Caflish (1994). See also Wang and Sloan (2008).

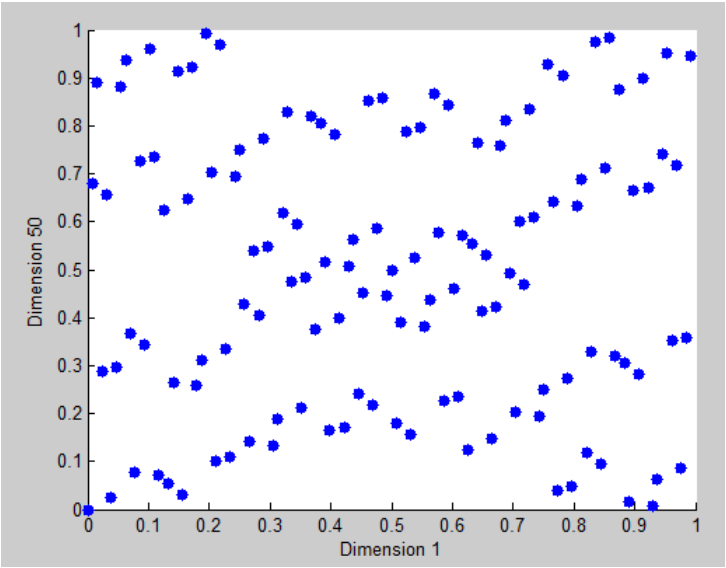

Figure 2.2: Dimension 1 vs Dimension 50 for $N = 128$

Hence, the low discrepancy points might perform badly when the dimensionality is increased.

2.2 Scrambling Techniques Under QMC

The finite parts (even for moderate sizes) in higher dimensions of many QMC sequences have poor properties, which can be alleviated using suitable scrambling techniques. Several methods are discussed later with the purpose of improving the uniformity of the QMC sequences in higher dimensions. Since most of these methods involve some form of scrambling of the coefficients in each of the radical inverse functions in an effort to redistribute the points of the sequence more uniformly, they are referred to as scrambling techniques. In the following paragraphs, we only introduce some popular scrambling methods, which are not discussed further in details.

Consider the unique digit expansion of a number n in base b as:

$$n = \sum_{j=0}^{\infty} a_j(n) b^j, where\ 0 \leq a_j(n) \leq b-1\ \ and\ b^j \leq n \leq b^{j+1}. \tag{2.5}$$

This unique expansion has only finitely many non-zero coefficients $a_j(n)$. The next step is to evaluate the radical inverse function in base b, which is defined as

$$\phi_b(n) = \sum_{j=0}^{\infty} a_j(n) b^{-j-1}. \tag{2.6}$$

Braaten and Weller (1979) described a permutation of the coefficients $a_j(n)$ that minimizes the discrepancy of the resulting scrambled Halton (1964) sequence. Their method suggests different permutations for different prime numbers, which effectively break the correlation across dimensions. Braaten and Weller have also proved that their scrambled sequence retains the theoretically appealing $O\left(N^{-1}\right)$ integration error of the standard Halton sequence.

The Random Digit scrambling approach (Matousek 1998) for Faure sequences is conceptually similar to the Braaten and Weller method. It suggests random permutations of the coefficients $a_j^k(n)$ to scramble the standard Faure sequence.

The Random Linear Scrambling approach (Matousek 1998) is based on the concept of cleverly introducing randomness in the recursive procedure of generating the coefficients for each successive dimension.

2.3 Randomization Strategies Under QMC

The fundamental property of QMC sequences is that they are deterministic sequences. They do not permit the practical estimation of integration error. Since a comparison of the performance of these sequences necessitates the computation of simulation variances and errors, it is necessary to randomize the QMC sequences. Randomization of QMC

sequences is a technique that introduces randomness into a deterministic QMC sequence while preserving the distribution property of the sequence.

2.3.1 Shifted Low Discrepancy Sequences

The shifted low discrepancy sequences are the most popular randomization strategy that is used recently. Suppose that $\boldsymbol{X}$ is a random variable uniformly distributed over $[0,1)^d$, and $\left(\boldsymbol{\varepsilon^{(n)}}\right)_{n\in N}$ be a low discrepancy sequence. Consider the random variable

$$\boldsymbol{Z} = \frac{1}{N}\sum_{n=1}^{N} f\left(\{\boldsymbol{X} + \boldsymbol{\varepsilon^{(n)}}\}\right) \tag{2.7}$$

instead of $\frac{1}{N}\sum\limits_{n=1}^{N} f\left(\boldsymbol{\varepsilon^{(n)}}\right)$ in QMC, where $\{\ \boldsymbol{x}\ \}$ is the vector of fractional parts of the coordinates of $\boldsymbol{x}$. The idea is to compute the average value of M independent copies $\boldsymbol{Z_i}$ of $\boldsymbol{Z}$ by

$$\frac{1}{M}\sum_{i=1}^{M} \boldsymbol{Z_i}, \tag{2.8}$$

and to obtain a confidence interval from the usual central limit theorem.

To compute f by the same number of times, we compare the variance of $\frac{1}{M}\sum\limits_{i=1}^{M} \boldsymbol{Z_i}$ with the variance of the standard MC simulation based on $M \times N$ randomly generated points. We will obtain a variance reduction if and only if the variance of QMC estimator is less than the variance of MC estimator. The convergence speed of the method has been shown in Tuffin (1997). If f is a function with bounded variation, the convergence speed of evaluating

$$\sigma^2\left(\frac{1}{N}\sum_{n=1}^{N} f\left(\boldsymbol{X} + \boldsymbol{\varepsilon^{(n)}}\right)\right)$$

is $O(N^{-2}\left(\log N\right)^{2d})$.

It is shown by Tuffin (1997), the convergence speed can even be faster for some special classes of functions. Let X be a random vector with uniform distribution over $[0,1]^d$. Then,

for all $\alpha > 1, C > 0$ and $N \leq 1$, there exists $g \in Z^d$ such that the convergence speed of evaluating

$$\max_{f \in E_\alpha^d(C)} \sigma^2 \left(\frac{1}{N} \sum_{n=0}^{N-1} f\left(\boldsymbol{X} + \frac{n}{N} g \right) \right) \tag{2.9}$$

is $O\left(N^{-2\alpha} (\ln N)^{2\alpha d}\right)$, where $E_\alpha^d(C)$ is the set of periodic functions $f : R^d \to R$ with period 1 over each coordinate, such that the Fourier coefficient of rank h of function f, $\hat{f}(h) = \int\limits_{[0,1]^d} f(x) e(-\langle h, x \rangle) dx$, where $\langle x, y \rangle$ is the standard inner product of $x, y \in R^d$ satisfies

$$|\hat{f}(h)| \leq C \left(\prod_{i=1}^{d} \max(1, |h_i|) \right)^{-\alpha} \quad \textit{for all } h \in Z^d.$$

2.3.2 Other Randomizations

Scrambled (t, d)-sequences is introduced by Owen (1995). The idea is to scramble the digits of special low discrepancy sequences, the (t, d)-sequences in base b by using random permutations for the digits, while preserving low discrepancy property.

Random-start Halton sequences views Halton sequence as an application of multidimensional von Neuman-Kakutani transformation (Okten 2009) with orbit vector $(0, ..., 0)^T$. By randomly choosing an orbit vector, they obtain randomized Halton sequences.

However, Tuffin (1997) showed that the randomized QMC might be as slow as MC. Nevertheless it was mentioned that even if the convergence rate is not always better, the variance is by a constant smaller and the simulation time decreases when using randomization. This is due to the fact that there are less calls to the pseudo-random or quasi-random number generators. For these two reasons, the randomization technique is still preferred.

2.4 Error Bound of QMC

According to Hickernell (1998), for QMC, if P is a low discrepancy point set, then the error bound takes the form

$$|I(f) - Q(f)| \leq D(P)V(f), \tag{2.10}$$

where $D(P)$ is the generalized discrepancy and $V(f)$ is the generalized variation of f. A special case of the above inequality is the well-known Koksma-Hlawka inequality, see Niederreiter (1992), which is defined as

$$|I(f) - Q(f)| \leq D^*(P)V(f), \tag{2.11}$$

where $D^*(P)$ is the traditional star-discrepancy and $V(f)$ is the variation of f on $[0,1]^d$ in the sense of Hardy and Krause.

Let $S = \{1, ..., d\}$ be the set of coordinate indices. For any $u \subseteq S$, let $|u|$ denote its cardinality, and let C^u denote the $|u|$-dimensional unit cube involving the coordinates in u. Furthermore, let x_u denotes the vector containing the components of x whose indices are in u, and let $dx_u = \prod_{j \in u} dx_j$ denotes the uniform measure on $C^u = [0,1)^u$.

Let $\|\|_p$ denote the L^p-norm of a function on C^u, that is

$$\|f\|_p = \left[\int_{C^u} |f|^p dx_u \right]^{1/p} \quad for\ 1 \leq p < \infty \tag{2.12}$$

$$and\ \|f\|_\infty = \inf\{\gamma \in C_u : |f| \leq \gamma\}. \tag{2.13}$$

This notation is extended to the case of a vector of functions (f_u), were u is an index running over some or all of the subsets of S, and each f_u is a function on C^u:

$$\|f_u\|_p = \left[\sum_u \|f\|_p^p \right]^{1/p} = \left[\sum_u \int_{C^u} |f_u|^p dx_u \right]^{1/p} \quad for\ 1 \leq p < \infty$$

and $\|f_u\|_\infty = \max_u \|f_u\|_\infty = \max_u \inf\{\gamma \in C_u : |f_u| \leq \gamma\}$.

Hickernell (1998) introduced a new notion of discrepancy, the so called L^p-star discrepancy, which is defined as

$$D_p^*(P) = \left\| \frac{|P \cap [0,x)|}{N} - v([0,x)) \right\|_p, \tag{2.14}$$

where v is the volume, and the L^p-star discrepancy is a natural generalization of $D^*(P)$. The appropriate definition of $D_p^*(P)$ is defined as

$$D_p^*(P) = \left\| \left(\frac{|P_u \cap [0,x_u)|}{N} - v([0,x_u)) \right)_{u \neq \phi} \right\|_p, \tag{2.15}$$

where P_u denotes the projection of the sample P into the cube C^u. This definition is also true for $p = \infty$. Then, one can generalize the upper bound formula to

$$|I(f) - Q(f)| \leq D_p^*(P) V_q(f) \ \textit{where} \ (p^{-1} + q^{-1} = 1), \tag{2.16}$$

and L^p-variation, $V_p(f)$, is a generalization of the variation in the sense of Hardy and Krause which can be written as follows:

$$V_p(f) = \left\| \left(\left. \frac{\partial^{|u|} f}{\partial x_u} \right|_{x_{S \setminus u} = (1,\ldots,1)} \right)_{u \neq \phi} \right\|_p. \tag{2.17}$$

For a certain broad class of integrands, Hickernell (1996) showed that

$$\begin{aligned} |I(f) - Q(f)| &\leq c|||f||| \ \textit{in general} \\ \textit{and} \ |I(f) - Q(f)| &\leq \tilde{c}|||f||| \ \textit{if } f \textit{ is periodic}, \end{aligned} \tag{2.18}$$

where $|||\ |||$ is a norm whose definition involves L^2-norms of the mixed partial derivatives of the function.

The primary mathematical tool used in Hickernell (1996) is reproducing kernel Hilbert spaces. It is also the main tool used here. Let $(X, \langle , \rangle)$ be some Hilbert space of real-valued functions on C^d, where $\langle , \rangle$ denotes the inner product. For any $x \in C^d$, let T_x denote the evaluation function as follows:

$$T_x(f) = f(x) \ \textit{for} \ \forall f \in X. \tag{2.19}$$

If T_x is bounded, the Cauchy-Schwarz inequality then implies the useful error bound:

$$|I(f) - Q(f)| = |\langle \zeta, f \rangle| \leq |||\zeta||| \; |||f||| \tag{2.20}$$

where the equality holds when f is a constant multiple of ζ. This means that ζ is the worst-case integrand. The quantity $|||\zeta|||$ only depends on the points P and may be identified as $D(P)$. Likewise $|||f|||$ can be identified as $V(f)$, a measure of the size or variation of the integrand.

Chapter 3

Effective Dimension

There is ample numerical evidence suggesting the relative efficiency of Quasi-Monte Carlo (QMC) methods when applied to finance applications, even if these problems are of several hundred dimensions. While the success of the QMC methods in finance applications cannot be fully explained by the Koksma-Hlawka error bound, see Niederreiter (1992), it is widely believed that it is attributed to the notion of effective dimension, as argued by Caflisch et al. (1997). In this chapter, we first introduce the concepts of dimension-wise quadrature, then we analyze the algorithm to determine the effective dimensions and show that the problems of determining their effective dimension is analytically tractable but computationally inefficient in many financial applications. A critical discussion of the impact of these techniques on the QMC error is presented. Section 3.1 reviews the Dimension-wise Quadrature, Section 3.2 introduces the dimension-wise decomposition, Section 3.3 describes the classical analysis of variance (ANOVA) decomposition, and Sections 3.3.1 and 3.3.2 discuss two types of effective dimensions: truncation dimension and superposition, respectively. In Section 3.4, we provide a new measure of effective dimension, which we denote as the delta dimension. Section 3.5 concludes the chapter by relating the effective dimensions, including our newly proposed delta dimension, to the QMC error bounds.

3.1 Dimension-wise Quadrature

High-dimensional integrals appear in various mathematical models in finance. In many cases, the arising integrals can not be calculated analytically and hence numerical methods must be applied. The curse of dimensionality, which states that the cost to compute an approximation with a prescribed accuracy depends exponentially on the dimension d of the problem, is one of the main obstacles for the numerical evaluation of high dimensional problems. In this section, we review the numerical methods for computing high-dimensional integrals.

Classical Multivariate Quadrature Methods

We start with the numerical methods for the computation of high-dimensional integrals

$$I(f) := \int_{[0,1]^d} f(x)dx \tag{3.1}$$

over the unit cube. Note that any rectangular integration domains $[a_1, b_1] \times ... \times [a_d, b_d]$ can be mapped to the unit cube by a simple linear transformation via

$$\int_a^b f(y)dy = (b-a)\int_0^1 f(a+(b-a)x)dx. \tag{3.2}$$

We also consider numerical methods for high-dimensional integrals of the form

$$I_\varphi(f) := \int_{R^d} f(\boldsymbol{x})\varphi_d(\boldsymbol{x})d\boldsymbol{x} \tag{3.3}$$

over the d-dimensional Euclidean space with the Gaussian weight function φ_d.

All numerical quadrature methods, which we discuss here, approximate the d-dimensional integral $I(f)$ by a weighted sum of n function evaluations

$$Q_N(f) := \sum_{i=1}^{N} w_i f(\boldsymbol{x^i}), \tag{3.4}$$

with weights $w_i \in R$ and nodes $\boldsymbol{x^i} = (x_1^i, ..., x_d^i) \in \Omega^d$. Here, Ω is either $[0,1]$ or R. Depending on the choice of the weights and nodes, different classes of methods with varying properties are obtained, for examples, QMC, MC, polynomial-based, and sparse grid methods. In this thesis, we only discuss and compare QMC and MC methods in the context of finance applications.

3.1.1 Crude Monte Carlo (MC) Methods

High dimensional integrals on the unit cube are mostly approximated with the MC method. Here, all weights equal to $w_i = 1/N$, and uniformly distributed sequences of pseudo-random numbers $\boldsymbol{x^i} \in [0,1]^d$ are used as simulation points. The law of large numbers then ensures that the estimate

$$MC(f) := \frac{1}{N} \sum_{i=1}^{N} f(\boldsymbol{x^i}) \tag{3.5}$$

converges to $I(f)$ for $N \to \infty$ if f has a finite variance, or in other words, $\sigma^2(f) = \int\limits_{[0,1]^d} (f(x) - I(f))^2 \, dx < \infty$. standard error of a MC estimate with N samples is approximately normally distributed with mean zero and standard deviation $\sigma(f)/\sqrt{N}$. A typical algorithm for obtaining a MC estimate and its standard error can be described as follow:

$$\hat{\sigma}_N(f) = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} [f(\boldsymbol{x^i}) - MC(f)]^2}. \tag{3.6}$$

The algorithm for the MC method can be described as follows:

Algorithm 3.1.1. *(MC Integration)*
Step 1: $I(f) = 0, q = 0$
Step 2: For $i = 1, ..., N$
Generate uniformly distributed random numbers $\boldsymbol{x}^i \in [0,1]^d$
$I(f) = I(f) + f(\boldsymbol{x}^i)$

End For

$MC(f) = I(f)/N$

Step 3: For $i = 1, ..., N$

$q = q + (f(x_i) - MC(f))^2$

End For

Step 4: $\sigma(f) = \sqrt{q/(N-1)}$

Step 5: return $MC(f)$, $\sigma(f)$

While the MC methods have a number of desirable properties, including their flexibilities and their convergence rates are independent of dimension, these methods are often being criticized for their slow rate of convergence. Hence, many variance reduction techniques have been proposed to increase the efficiency of MC methods. Details of these methods can be found in say Glasserman (2004). The quasi-Monte Carlo methods, which we will discuss in the next subsection, is another sampling-based method alternate to MC in an attempt to outperform the MC methods.

3.1.2 Quasi-Monte Carlo (QMC) Methods

QMC methods are also another type of equal-weight sampling methods for evaluating high-dimensional integrals, just like MC. Instead of using pseudo-random numbers as in the MC methods, QMC methods rely on specially constructed sequences known as the low discrepancy sequences. The low discrepancy sequences are deterministic and are known to have greater uniformity than the traditional pseudo-random sequences. As we have mentioned in the last chapter, the irregularity or the uniformity of a set of points in the unit cube is measured by the discrepancy. Some popular low discrepancy sequences are attributed to Halton (1960), Faure (1982), and Sobol (1967). These sequences have also been extensively used in finance applications (see Lemieux, 2009).

The worst case error bound associated with the method of QMC can be obtained from the

Koksma-Hlawka inequality which shows that the worst case error of a QMC method with N samples is of order

$$\epsilon(N) = O\left(N^{-1}(\log N)^d\right) \tag{3.7}$$

for integrands of bounded variation. This rate is asymptotically better than the probabilistic $O(N^{-1/2})$ error bound of MC.

In practical applications, QMC methods involving randomized low discrepancy sequences, as opposed to the traditional deterministic low discrepancy sequences, are typically implemented. As we have discussed briefly that the traditional deterministic low discrepancy sequences can be randomized by say random shifts or random permutations of digits. The randomized QMC permits an easy way of obtaining a statistical error estimate while maintaining the higher convergence rate of QMC. For a survey on the randomized QMC, see Lecuyer and Lemieux (2002).

3.2 Dimension-wise Decomposition

Griebel and Holtz (2010) first introduced the general dimension-wise decompositions with applications to finance. Let $\Omega \subseteq R$ be a set and let

$$d\mu(\boldsymbol{x}) = \prod_{j=1}^{d} d\mu_j(x_j) \tag{3.8}$$

denote a d-dimensional product measure defined on Borel subsets of Ω^d. Here, $\boldsymbol{x} = (x_1, ..., x_d)^T$ and μ_j for $j = 1, ..., d$ are probability measures on Borel subsets of Ω. Let $V^{(d)}$ be the Hilbert space of all functions $f : \Omega^d \to R$ with the inner product

$$\langle f, g\rangle := \int_{\Omega^d} f(\boldsymbol{x})g(\boldsymbol{x})d\mu(\boldsymbol{x}). \tag{3.9}$$

For a given set $u \subseteq D$, where $D := \{1, ..., d\}$ denotes the set of coordinate indices, the measure μ induces projections $P_u : V^{(d)} \to V^{(|u|)}$ by

$$P_u f(\boldsymbol{x_u}) := \int_{\Omega^{d-|u|}} f(\boldsymbol{x})d\mu_{D\setminus u}(\boldsymbol{x}), \tag{3.10}$$

where $\boldsymbol{x_u}$ denotes the $|u|-$dimensional vector containing those components of x whose indices belong to the set u and $d\mu_{D\setminus u}(\boldsymbol{x}) := \prod_{j\notin u} d\mu_j(\boldsymbol{x_j})$. In the case $\boldsymbol{u} = \emptyset$ is defined as

$$P_\emptyset f(\boldsymbol{x_\emptyset}) := \int_{\Omega^d} f(\boldsymbol{x}) d\mu(\boldsymbol{x}) =: I(f). \tag{3.11}$$

The projections define a dimension-wise decomposition of $f \in V^{(d)}$ into a finite sum according to

$$f(x_1, ..., x_d) = f_0 + \sum_{i=1}^{d} f_i(x_i) + \sum_{i,j=1, i<j}^{d} f_{i,j}(x_i, x_j) + ... + f_{1,...,d}(x_1, ..., x_d), \tag{3.12}$$

which is often written in a more compact notation as follows:

$$f(\boldsymbol{x}) = \sum_{u\subseteq D} f_u(\boldsymbol{x_u}). \tag{3.13}$$

The 2^d terms f_u describe the dependence of the function f on the dimensions $j \in u$ with respect to the measure μ. They could be defined recursively by

$$f_u(\boldsymbol{x_u}) := P_u f(\boldsymbol{x_u}) - \sum_{v\subset u} f_v(\boldsymbol{x_v}), \tag{3.14}$$

or equivalently given by

$$f_u(\boldsymbol{x_u}) = \sum_{v\subseteq u} (-1)^{|u|-|v|} P_v f(\boldsymbol{x_v}). \tag{3.15}$$

The resulting decomposition is unique for a fixed measure μ and orthogonal in the sense that

$$\langle f_u, f_v \rangle = 0. \tag{3.16}$$

3.3 Classical ANOVA Decomposition

ANOVA decomposition is a way of decomposing a function into a sum of simpler functions. Let $D = \{1, ..., d\}$. For any subset $u \subseteq D$, let $|u|$ denotes its cardinality and $D\setminus u$ denotes its

complementary set in D. Let $\boldsymbol{x_u}$ be the $|u|-$dimensional vector containing the coordinates of $\boldsymbol{x}$ with indices in u. Furthermore, let C^u denotes the $|u|$-dimensional unit cube involving the coordinates in u.(i.e. $C^d = [0,1)^d$). Assume that $f(\boldsymbol{x})$ is a square integrable function. We can write $f(x)$ as the sum of its 2^d ANOVA terms:

$$f(\boldsymbol{x}) = \sum_{u \subseteq D} f_u(\boldsymbol{x}). \tag{3.17}$$

The ANOVA terms $f_u(\boldsymbol{x})$ are defined recursively by

$$f_u(\boldsymbol{x}) = \int_{C^{d-|u|}} f(\boldsymbol{x}) d\boldsymbol{x_{D \setminus u}} - \sum_{v \subset u} f_v(\boldsymbol{x}) \tag{3.18}$$

with $f_\emptyset = \int_{C^d} f(\boldsymbol{x})dx = I(f)$. The ANOVA term $f_u(\boldsymbol{x})$ is the part of the function depending only on the variables x_j with $j \in u$. The ANOVA terms enjoy some interesting properties:

1. $\int_0^1 f_u(\boldsymbol{x})dx_j = 0$ *for* $j \in u$.

2. The ANOVA decomposition is orthogonal whenever $u \neq v$. i.e. $\int_{C^d} f_u(\boldsymbol{x}) f_v(\boldsymbol{x}) d\boldsymbol{x} = 0$.

3. The orthogonality implies that $\sigma^2(f) := \int_{C^d} (f(\boldsymbol{x}) - I(f))^2 dx = \int_{C^d} f^2(\boldsymbol{x}) d\boldsymbol{x} - [I(f)]^2$ be the variance of the function f, and also it can be written as

$$\sigma^2(f) = \sum_{u \subseteq D} \sigma_u^2(f), \tag{3.19}$$

   where $\sigma_u^2(f) = \int_{c^d} [f_u(\boldsymbol{x})]^2$ for $|u| > 0$ is the variance of f_u, and $\sigma_\emptyset^2(f) = 0$.

Sobol' and Kucherenko (2005) introduced the value $\sigma_u^2(f)/\sigma^2(f)$, called global sensitivity indices, which can be used to measure the relative importance of the term f_u with respect to the function f.

To be more precise, let u be a subset of D, the variance corresponding to u is defined as

$$D_u(f) = \sum_{v \subseteq u} \sigma_v^2(f). \tag{3.20}$$

The total-effect variance corresponding to u is defined as

$$D_u(f^{tot}) = \sum_{v \cap u \neq \emptyset} \sigma_v^2(f) = \sigma^2(f) - \sigma_{D \setminus u}^2(f). \tag{3.21}$$

The total-effect variance $D_u(f^{tot})$ represents the total contribution of the variable x_u to the variability of the function f. It includes both the pure effect and the effects due to the interactions between x_u and x_v, where $v \cap u \neq \emptyset$.

The size of $D_u(f)$ and $D_u(f^{tot})$ determine the relative importance of x_u. If $\sigma^2(f)$ is close to $D_u(f)$, then the pure effect of x_u affects the function, and its interactions with others could be ignored. If $D_u(f)$ is small but $D_u(f^{tot})$ is large, then the interactions of x_u with others affect the function, and the pure effect could be ignored. If $D_u(f^{tot})$ is small, x_u has relatively small effect on the function f.

Using the ANOVA decomposition, Caflish et al. (1997) give two definitions of effective dimensions, known as the truncation dimension and the superposition dimension. These are discussed in the subsequent two subsections.

3.3.1 Truncation Dimension

Definition 3.3.1. *The effective dimension of f in the truncation sense (or truncation dimension) is the smallest integer d_t such that*

$$\sum_{u \subseteq \{1,\ldots,d_t\}, u \neq \emptyset} \sigma_u^2(f) \geq \alpha \sigma^2(f), \tag{3.22}$$

where $\alpha \in (0, 1)$.

In the above definition, the parameter α is arbitrary but it is often set to some values close to one, such as 0.99. Roughly speaking, the truncation dimension represents that the first d_t variables are "important" variables of the function f. For large d, it is no longer possible to compute all 2^d ANOVA terms, but the truncation dimension can be computed

recursively by

$$D_u(f) = \int_{[0,1)^{2d-|u|}} f(x)f(x_u, y_{D\setminus u})dxdy_{D\setminus u} - (I(f))^2, \tag{3.23}$$

where $D_u(f)$ is computed by $(2d-|u|)$-dimensional integral.

The algorithm of computing the truncation dimension proposed by Barth et al. (2011) is given as follow:

Algorithm 3.3.1. *(truncation dimension):*
Step 1: Compute $I(f)$ *and* $\sigma^2(f)$
Step 2: For $i = 1, , , d$
Compute $D_{1,\dots,i}(f)$ *by* (3.23).
If $D_{1,\dots,i}(f) > \alpha\sigma^2(f)$, *then*
return i
End If
End For

We can see from the algorithm that the more precise the integrals (3.23) can be computed, the better is the estimation of the truncation dimension. Overall the algorithm requires to compute at most $d+1$ integrals with each integral of dimension up to $2d-1$.

With some additional efforts, the algorithm could be also written as

$$T_i = \frac{1}{\sigma^2(f)} \sum_{u \not\subseteq \{1,\dots,i\}} \sigma_u^2(f) \text{ for i=0,1...,d.} \tag{3.24}$$

The value T_i denotes the percentage of the variance which is not explained by the leading i dimensions. In particular, $T_0 = 1$ and $T_d = 0$. The decay of the values $T_1 - T_0, ..., T_d - T_{d-1}$ illustrates the importance of the variables x_i, for $i = 1, ..., d$, which is independent of the proportion α.

3.3.2 Superposition Dimension

Definition 3.3.2. *For $\alpha \in (0,1)$, the effective dimension of f in the superposition sense (or superposition dimension) is the smallest integer d_s such that*

$$\sum_{|u|\leq d_s, u\neq\emptyset} \sigma_u^2(f) \geq \alpha\sigma^2(f). \tag{3.25}$$

The superposition dimension roughly represents the highest order of important interactions between variables. The computation of superposition dimension needs to compute the values $\sigma_u^2(f)$, for $u \in \{1,...,d\}$, calculated from (3.23) in the order that if $d_s = 1$, we compute $\binom{d}{1}$ values; if $d_s = 2$, we compute $\binom{d}{1} + \binom{d}{2}$ values, and etc. So, they are computationally inefficient even when d_s is small, e.g. $d_s \geq 3$.

The algorithm of computing the superposition dimension proposed by Barth et al. (2011) is given as follow:

Algorithm 3.3.2. *(superposition dimension):*
Step 1: Compute $I(f)$ and $\sigma^2(f)$, $D\left(f^{tot}\right) = 0$
Step 2: For $i = 1,...,d$
Compute $\sigma_i^2(f) = D_i(f)$ by (3.23).
$\sigma^2(f^{tot}) = \sigma^2(f^{tot}) + \sigma_i^2(f)$
End For
Step 3: If $\sigma^2\left(f^{tot}\right) > \alpha\sigma^2(f)$
return $d_s = 1$
End If
Step 4: For $i = 1,...,d$
For $j = i+1,...,d$
Compute $D_{i,j}(f)$ by (3.23).
$\sigma_{i,j}^2(f)$ = $D_{i,j}(f)$ - $\sigma^2(f_i)$ - $\sigma^2(f_j)$
$\sigma^2\left(f^{tot}\right) = \sigma^2\left(f^{tot}\right) + \sigma^2(f_{i,j})$

End For

End For

Step 5: If $\sigma^2\left(f^{tot}\right) > \alpha\sigma^2(f)$

return $d_s = 2$

Else

return $d_s \geq 3$

End If

With some additional efforts, the algorithm could be also written as

$$S_i = \frac{1}{\sigma^2(f)} \sum_{|u|>i} \sigma^2(f_u) \text{ for i=0,1...,d.} \tag{3.26}$$

The value S_i denotes the percentage of the variance which is not explained by the leading i dimensions. In particular, $S_0 = 1$ and $S_d = 0$. The value S_i illustrates the variability of all subsets of i variables chosen from $\{x_1, ..., x_d\}$, for $i = 1, 2..., d$, which is independent of the proportion α.

Clearly, the proposed algorithms only computes the variance contributions of the order-1 and order-2 terms of the ANOVA decomposition. The algorithms can be generalized to higher orders but the computational cost grows exponentially in the superposition dimension d_s. i.e. The computation of the superposition dimension is only feasible for lower-dimensional functions or functions with "very low" superposition dimensions.

3.4 Delta dimension

In many of the financial problems, the truncation dimension can be quite high, and we have seen that calculating either truncation dimension or superposition dimension is computationally burdensome when the underlying effective is high. In this section, we propose a method to calculate so called delta dimension as follows.

Suppose that $\boldsymbol{x} = (x_1, ..., x_d)^T \sim N_d(\mathbf{0}, \boldsymbol{\Sigma})$, and $f(\boldsymbol{x})$ can be written in the form $f(\boldsymbol{x}) = f_1(\boldsymbol{x}) + f_2(\boldsymbol{x}) + \cdots + f_d(\boldsymbol{x})$. We then let $Y = \{f_1(\boldsymbol{x}),\ f_2(\boldsymbol{x})\ ,\ ...\ ,\ f_d(\boldsymbol{x})\}$. By applying the first-order vector Taylor expansion to each of the function f_i, for $i = 1, ..., d$, around an arbitrary vector $\boldsymbol{x} = \hat{\boldsymbol{x}} + \Delta\boldsymbol{x}$:

$$\boldsymbol{Y} \approx \boldsymbol{Y}\left(\hat{\boldsymbol{x}}\right) + \nabla\boldsymbol{Y}\left(\hat{\boldsymbol{x}}\right)\left(\boldsymbol{x} - \hat{\boldsymbol{x}}\right). \tag{3.27}$$

Then, applying the Delta method, the covariance matrix of $\boldsymbol{Y}$ is defined as follows:

$$Var(\boldsymbol{Y}) = \boldsymbol{J}\boldsymbol{\Sigma}\boldsymbol{J}^T \tag{3.28}$$

where $\boldsymbol{J}$ is the Jacobian matrix of $\boldsymbol{Y}$ at $\boldsymbol{x}$; i.e.

$$\boldsymbol{J} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_d} \\ \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_2}{\partial x_d} \\ \cdots & \cdots & \cdots \\ \frac{\partial f_d}{\partial x_1} & \cdots & \frac{\partial f_d}{\partial x_d} \end{pmatrix}.$$

Suppose there exists a decomposition matrix $\boldsymbol{A}$ such that $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{\Sigma}$, and we define $\hat{\boldsymbol{A}}_{\boldsymbol{k}}$ as the first k columns of the decomposition matrix $\boldsymbol{A}$, and $\hat{\boldsymbol{A}}_{\boldsymbol{d-k}}$ as its remaining columns for $k = 1, 2, ..., d$. Suppose further that $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{z}$ and $\hat{\boldsymbol{x}} = \hat{\boldsymbol{A}}_{\boldsymbol{k}}\hat{\boldsymbol{z}}_{\boldsymbol{k}}$, where $\boldsymbol{z} \sim N_d(\mathbf{0}, \boldsymbol{I})$ and $\hat{\boldsymbol{z}}_{\boldsymbol{k}}$ is the first k elements of $\boldsymbol{z}$. We have the following two lemmas:

Lemma 3.4.1. *$\hat{\boldsymbol{A}}_{\boldsymbol{k}}\hat{\boldsymbol{A}}_{\boldsymbol{k}}{}^T$ and $\hat{\boldsymbol{A}}_{\boldsymbol{d-k}}\hat{\boldsymbol{A}}_{\boldsymbol{d-k}}{}^T$ are both positive semi-definite for $\forall k = 1, ..., d$.*

Proof: For any 1 by d row vector $\boldsymbol{y}$, we have

$$\boldsymbol{y}\hat{\boldsymbol{A}}_{\boldsymbol{k}}\hat{\boldsymbol{A}}_{\boldsymbol{k}}{}^T\boldsymbol{y}^T = \boldsymbol{y_2}\boldsymbol{y_2}{}^T = \sum_{i=1}^{k}(\boldsymbol{y_2})_i^2 \geq 0,$$

where $\boldsymbol{y_2} = \boldsymbol{y}\hat{\boldsymbol{A}}_{\boldsymbol{k}}$ is a 1 by k vector. So $\hat{\boldsymbol{A}}_{\boldsymbol{k}}$ is positive semi-definite. Similarly, for $\hat{\boldsymbol{A}}_{\boldsymbol{d-k}}\hat{\boldsymbol{A}}_{\boldsymbol{d-k}}{}^T$ except that $\boldsymbol{y_2} = \boldsymbol{y}\hat{\boldsymbol{A}}_{\boldsymbol{d-k}}$ is a 1 by $d - k$ vector. □

Lemma 3.4.2. *If a d by d matrix $\boldsymbol{H}$ is positive semi-definite, then the sum of all of its elements is greater than or equal to 0.*

(Proof is omitted since it's clear to see.)

Lemma 3.4.3. $\boldsymbol{A}\boldsymbol{A}^T = \hat{\boldsymbol{A}}_{\boldsymbol{k}}\hat{\boldsymbol{A}}_{\boldsymbol{k}}{}^T + \hat{\boldsymbol{A}}_{\boldsymbol{d-k}}\hat{\boldsymbol{A}}_{\boldsymbol{d-k}}{}^T$, *for* $\forall k = 1, ..., d$.

(Proof is omitted since it's clear to see.)

Then, our delta dimension is defined as follows:

Definition 3.4.1. *(Delta Dimension)*
The delta dimension we define here is

$$\begin{aligned} d_d &= \min\{k : Var(f_1(\hat{\boldsymbol{x}}) + ... + f_d(\hat{\boldsymbol{x}})) \\ &= \sum_{i=1}^{d}\sum_{j=1}^{d}\left(\boldsymbol{J}\hat{\boldsymbol{A}}_{\boldsymbol{k}}\hat{\boldsymbol{A}}_{\boldsymbol{k}}{}^T\boldsymbol{J}^T\right)_{ij} \qquad (3.29) \\ &\geq \alpha\sum_{i=1}^{d}\sum_{j=1}^{d}\left(\boldsymbol{J}\boldsymbol{\Sigma}\boldsymbol{J}^T\right)_{ij} \\ &= \alpha Var(f_1(\boldsymbol{x}) + f_2(\boldsymbol{x}) + \cdots + f_d(\boldsymbol{x})) \\ &= \alpha Var(f)\} \qquad (3.30) \end{aligned}$$

where $(\boldsymbol{X})_{ij}$ *means the element in* i^{th}*row and* j^{th}*column of matrix* $\boldsymbol{X}$*; i.e. the formula* (3.29) *is equivalent to find the minimum* k *such that* $\sum_{i=1}^{d}\sum_{j=1}^{d}\left(\boldsymbol{J}\hat{\boldsymbol{A}}_{\boldsymbol{k}}\hat{\boldsymbol{A}}_{\boldsymbol{k}}{}^T\boldsymbol{J}^T\right)_{ij} \geq \alpha\sum_{i=1}^{d}\sum_{j=1}^{d}\left(\boldsymbol{J}\boldsymbol{\Sigma}\boldsymbol{J}^T\right)_{ij}$.

Remark 3.4.1. *Since by Lemma 3.4.3, we have*

$$\boldsymbol{J}\boldsymbol{\Sigma}\boldsymbol{J}^T = \boldsymbol{J}\hat{\boldsymbol{A}}_{\boldsymbol{k}}\hat{\boldsymbol{A}}_{\boldsymbol{k}}{}^T\boldsymbol{J}^T + \boldsymbol{J}\hat{\boldsymbol{A}}_{\boldsymbol{d-k}}\hat{\boldsymbol{A}}_{\boldsymbol{d-k}}{}^T\boldsymbol{J}^T$$

Then, by Lemma 3.4.1, we know that the matrices $\boldsymbol{J}\boldsymbol{\Sigma}\boldsymbol{J}^T$, $\boldsymbol{J}\hat{\boldsymbol{A}}_{\boldsymbol{k}}\hat{\boldsymbol{A}}_{\boldsymbol{k}}{}^T\boldsymbol{J}^T$ *and* $\boldsymbol{J}\hat{\boldsymbol{A}}_{\boldsymbol{d-k}}\hat{\boldsymbol{A}}_{\boldsymbol{d-k}}{}^T\boldsymbol{J}^T$ *are all positive semi-definite, then by Lemma 3.4.2, we have*

$$\sum_{i=1}^{d}\sum_{j=1}^{d}(\boldsymbol{J}\boldsymbol{\Sigma}\boldsymbol{J})_{ij} \geq \sum_{i=1}^{d}\sum_{j=1}^{d}(\boldsymbol{J}\hat{\boldsymbol{A}}_{\boldsymbol{k}}\hat{\boldsymbol{A}}_{\boldsymbol{k}}{}^T\boldsymbol{J}^T)_{ij} \geq 0. \qquad (3.31)$$

We denote the minimum k that satisfies the above inequality as the *delta effective dimension*, and the delta dimension we mean here is the minimum number of columns we need for an arbitrary decomposition to maintain the variability larger than $\alpha Var(f)$. The advantage of calculating the delta effective dimension is that there is no integration evaluation and the calculation is computationally efficient. The diagonal elements of the matrix $\boldsymbol{J\Sigma J}^T$ represent the importance of decomposed functions f_i's, for $i = 1, ..., d$. In the special case of $f = f_1(x_1) + f_2(x_2) + ... + f_d(x_d)$, the diagonal elements of $\boldsymbol{J\Sigma J}^T$ also represents the importance of x_i's, for $i = 1, ..., d$. The QMC error bound of this form is hard to determine due to the correlation of f_i and f_j for $\forall i \neq j$, however, from the covariance matrix of the functions f_i, we could find a good measure of $\|f_i\|$, for $i = 1, ..., d$.

Remark 3.4.2. *Note that the delta dimension will be changed if we change the structure of the decomposition of the function f, which will be discussed in Section 4.2.*

Remark 3.4.3. *Note that $\hat{\boldsymbol{x}} = \hat{\boldsymbol{A}}_{\boldsymbol{k}}\hat{\boldsymbol{z}}_{\boldsymbol{k}}$ is a d by 1 vector, which defines the number of $U(0,1)$ random variables we need in order to approximate the multivariate normal random variable $\boldsymbol{x}$. The minimum number of $U(0,1)$ random variables we need to get the variability of $f(\hat{\boldsymbol{x}})$ larger than α times the variability of $f(\boldsymbol{x})$ with an arbitrary decomposition matrix $\boldsymbol{A}$ is defined as the delta dimension. In other words, the delta dimension depends on the selection of the matrix $\hat{\boldsymbol{A}}_{\boldsymbol{k}}$. i.e. the first k columns of the selected decomposition matrix $\boldsymbol{A}$. The optimal selection methods of the decomposition matrix $\boldsymbol{A}$ will be further discussed in Section 4.2.*

3.5 Error Bound

The QMC error depends on the effective dimension. Wang and Fang (2003) had shown that under appropriate definitions of discrepancy and variation, the error bound can be written as

$$|Q(f) - I(f)| \leq \sum_{u \subseteq D} D_u(P_u)\|f_u\|, \tag{3.32}$$

where P_u is the projection of the point set P on C^u, $D_u(P_u)$ is the discrepancy of P_u and $\|f_u\|$ is the variation of f_u. i.e. this equation shows that QMC error depends on the uniformity of all projections and all low-dimensional f_u's.

The biggest advantage of low discrepancy points is that they have better uniformity than random points, but as we have seen in the previous Chapter, this better uniformity can not be preserved for all dimensions. So, understanding the following possible advantage and potential problem of QMC is crucial to solve financial problems.

1. At least the first few dimensions of low discrepancy points have better uniformity than random points. In other word, for small l, $D_u(P_u)$ with $u \subseteq \{1, ..., l\}$, low discrepancy points perform better than random points. But for other subsets, the results could be worse than those of random points, unless the number of points is extremely large.

2. The low-order projections of low discrepancy points have better performance "on the average" than random points. Many low discrepancy sequences have poor projections even in small l. In fact, even some two-dimensional sequences have been observed by Wang and Fang (2003) with some poor projections for some small l. It had also been shown that for $d \in [10, 100]$, and l is small (say $l \leq 3$), the superposition discrepancy of low discrepancy set is smaller than that of random points, where we defined the superposition discrepancy as

$$D_{(l)}(P) = \left(\sum_{|u|=l} (D_u(P_u))^2 \right)^{1/2}. \tag{3.33}$$

   Furthermore, the superiority decreases as l and d increase. For $l > 3$ and $d > 30$, the superposition discrepancies of low discrepancy points and random points are almost the same, unless the number of points is huge. Thus, in this situation considering the structure of the function f is more important than the projections onto low discrepancy points, which will be discussed in Chapter 4.

As argued in Wang and Fang (2003), the error bound of the function f under truncation

dimension d_t can be defined as

$$|Q(f) - I(f)| \leq \sum_{u \subseteq \{1,\ldots,d_t\}} D_u(P_u)\|f_u\| + \sum_{u \cap (D \setminus \{1,\ldots,d_t\}) \neq \emptyset} D_u(P_u)\|f_u\|. \tag{3.34}$$

If d_t is small (say $d_t \leq 10$), the first term on the right hand side of (3.34) is much smaller for QMC than for MC. Considering $\|f_u\|$ of the second term on the right hand side of (3.34) is usually small, so all terms are expected to be smaller for QMC than for MC. Thus, if f has low truncation dimension, QMC is usually better than MC. However, if d_t is large, considering first few leading coordinates of low discrepancy points, and minimizing $\|f_u\|$ for the rest of dimensions have the potential to improve the QMC results.

The error bound of the function f under superposition dimension d_s can be defined as

$$\begin{aligned} |Q(f) - I(f)| \quad &\leq \sum_{|u| \leq d_s} D_u(P_u)\|f_u\| + \sum_{|u| > d_s} D_u(P_u)\|f_u\| \\ &\leq \sum_{l=1}^{d_s} D_l(P) \left(\sum_{|u|=l} \|f_u\|^2 \right)^{1/2} + \sum_{|u| > d_s} D_u(P_u)\|f_u\|. \end{aligned} \tag{3.35}$$

If d_s is small (say $d_s \leq 3$), then the superposition discrepancy of low discrepancy points is smaller than random points. Thus, if f has low superposition dimension, QMC is usually better than MC. So when d_s is large, minimizing $\|f_u\|$ on the second term is crucial, and this is also the motivation of our proposed method that we will discuss in the next chapter.

Furthermore, the following three lemmas relate the effective dimensions to approximation errors. The parameter $\alpha \in [0, 1]$ and is typically set to a value close to one. The first two lemmas can be found in Barth et al. (2011) and the last lemma, which is based on our newly defined delta dimension, is new. We have included the proofs to the first two lemmas for completeness.

Lemma 3.5.1. *Suppose the function f has the truncation dimension d_t with proportion α and let $f_{d_t} = \sum_{u \subseteq \{1,\ldots,d_t\}} f_u(x_u)$. Then*

$$\|f - f_{d_t}\|_{L_2}^2 \leq (1 - \alpha)\sigma^2(f). \tag{3.36}$$

Proof: Since we have $\sigma^2(f_u) = \|f_u\|_2^2$ for $u \neq \emptyset$ and $\int\limits_{[0,1)^{|u|}} f_u(x_u)dx_u$ for $u \neq \emptyset$. From the equation $f(x) = \sum\limits_{u \subseteq D} f_u(x_u)$, we have

$$\begin{aligned} \|f - f_{d_t}\|_{L_2}^2 &= \left\| \sum_{u \subsetneq \{1,\ldots,d_t\}} f_u \right\|_{L_2}^2 \\ &= \sum_{u \subsetneq \{1,\ldots,d_t\}} \|f_u\|_{L_2}^2 \\ &= \sum_{u \subseteq D} \sigma^2(f_u) - \sum_{u \subseteq \{1,\ldots,d_t\}} \sigma^2(f_u) \\ &\leq (1-\alpha)\sigma^2(f). \end{aligned}$$

□

Lemma 3.5.2. *Suppose the function f has the superposition dimension d_s with proportion α and let $f_{d_s}(x) = \sum\limits_{|u| \leq d_s} f_u(x_u)$. Then*

$$\|f - f_{d_s}\|_{L_2}^2 \leq (1-\alpha)\sigma^2(f). \tag{3.37}$$

Proof: Similar to Lemma 3.5.1, we have

$$\begin{aligned} \|f - f_{d_s}\|_{L_2}^2 &= \left\| \sum_{|u| > d_s} f_u \right\|_{L_2}^2 \\ &= \sum_{|u| > d_s} \|f_u\|_{L_2}^2 \\ &= \sum_{|u| > d_s} \sigma^2(f_u) \\ &\leq (1-\alpha)\sigma^2(f). \end{aligned} \tag{3.38}$$

□

Lemma 3.5.3. *Suppose the function f has the delta dimension d_d with proportion α. Let $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{z}$, for $\boldsymbol{z} \sim N_d(\boldsymbol{0}, \boldsymbol{I})$, and $\hat{\boldsymbol{x}} = \hat{\boldsymbol{A}}_{\boldsymbol{d_d}} \hat{\boldsymbol{z}}_{\boldsymbol{d_d}}$, where $\hat{\boldsymbol{A}}_{\boldsymbol{d_d}}$ is first $d - d_d$ columns of $\boldsymbol{A}$, and*

$\hat{\boldsymbol{z}}_{\boldsymbol{d_d}}$ is the vector of first d_d random variables of $\boldsymbol{z}$. We also define

$$\tilde{\boldsymbol{x}} = \hat{\boldsymbol{A}}_{\boldsymbol{d-d_d}} \hat{\boldsymbol{z}}_{\boldsymbol{d-d_d}},$$

where $\hat{\boldsymbol{A}}_{\boldsymbol{d-d_d}}$ is remaining $d-d_d$ columns of $\boldsymbol{A}$, and $\hat{\boldsymbol{z}}_{\boldsymbol{d-d_d}}$ is the vector of remaining $d-d_d$ random variables of $\boldsymbol{z}$. Obviously, $\boldsymbol{x}$, $\hat{\boldsymbol{x}}$ and $\tilde{\boldsymbol{x}}$ are all d by 1 vectors.

We further suppose that $f_{d_d}(\boldsymbol{x}) = \sum_{i=1}^{d} f_i(\boldsymbol{x})$ such that the covariance of $\{f_1(\boldsymbol{x}),..., f_d(\boldsymbol{x})\}$ has the form $\boldsymbol{J\Sigma J}^T$, where $\boldsymbol{J}$ is the Jacobian matrix of $\{f_1, ..., f_d\}$ with respect to $\{x_1, ..., x_d\}$ and $\boldsymbol{x}$ has mean $\mathbf{0}$ and covariance matrix $\boldsymbol{\Sigma}$. Then

$$Var(f(\tilde{\boldsymbol{x}})) \leq (1-\alpha)\sigma^2(f). \tag{3.39}$$

Proof: Since $f(\tilde{\boldsymbol{x}}) = f(\hat{\boldsymbol{A}}_{\boldsymbol{d-d_d}} \hat{\boldsymbol{z}}_{\boldsymbol{d-d_d}})$, by (3.28), we have

$$f(\tilde{\boldsymbol{x}}) \sim N_{d-d_d}(\mathbf{0}, \boldsymbol{J}\hat{\boldsymbol{A}}_{\boldsymbol{d-d_d}} \hat{\boldsymbol{A}}_{\boldsymbol{d-d_d}}{}^T \boldsymbol{J}^T).$$

By Lemma 3.5.3, we can write this as

$$f(\tilde{\boldsymbol{x}}) \sim N_{d-d_d}(\mathbf{0}, \boldsymbol{J}(\boldsymbol{AA}^T - \hat{\boldsymbol{A}}_{\boldsymbol{d_d}} \hat{\boldsymbol{A}}_{\boldsymbol{d_d}}{}^T)\boldsymbol{J}^T).$$

Then,

$$\begin{aligned} Var(f(\tilde{\boldsymbol{x}})) &= \sum_{i=1}^{d}\sum_{j=1}^{d} (\boldsymbol{J}(\boldsymbol{AA}^T - \hat{\boldsymbol{A}}_{\boldsymbol{d_d}} \hat{\boldsymbol{A}}_{\boldsymbol{d_d}}{}^T)\boldsymbol{J}^T)_{ij} \\ &= \sum_{i=1}^{d}\sum_{j=1}^{d} (\boldsymbol{J}(\boldsymbol{AA}^T)\boldsymbol{J}^T)_{ij} - \sum_{i=1}^{d}\sum_{j=1}^{d} (\boldsymbol{J}(\hat{\boldsymbol{A}}_{\boldsymbol{d_d}} \hat{\boldsymbol{A}}_{\boldsymbol{d_d}}{}^T)\boldsymbol{J}^T)_{ij}. \end{aligned} \tag{3.40}$$

According to the Definition 3.4.1, we have

$$\sum_{j=1}^{d} \left(\boldsymbol{J}\hat{\boldsymbol{A}}_{\boldsymbol{d_d}} \hat{\boldsymbol{A}}_{\boldsymbol{d_d}}{}^T \boldsymbol{J}^T\right)_{ij} > \alpha \sum_{i=1}^{d}\sum_{j=1}^{d} \left(\boldsymbol{J\Sigma J}^T\right)_{ij}.$$

Then, the Equation (3.40) becomes

$$\begin{aligned} Var(f(\tilde{\boldsymbol{x}})) &\leq \sum_{i=1}^{d}\sum_{j=1}^{d}(\boldsymbol{J}(\boldsymbol{A}\boldsymbol{A}^T)\boldsymbol{J}^T)_{ij} - \alpha\sum_{i=1}^{d}\sum_{j=1}^{d}\left(\boldsymbol{J}\boldsymbol{\Sigma}\boldsymbol{J}^T\right)_{ij} \\ &= (1-\alpha)\sum_{i=1}^{d}\sum_{j=1}^{d}(\boldsymbol{J}(\boldsymbol{A}\boldsymbol{A}^T)\boldsymbol{J}^T)_{ij} \\ &= (1-\alpha)Var(f). \end{aligned}$$

□

Remark 3.5.1. *In our decomposed functional structures $f = f_1(\boldsymbol{x}) + \cdots + f_d(\boldsymbol{x})$, f_i and f_j are correlated and non-orthogonal to each other, for $i \neq j$. So, we can not apply the L_2 normality of the functions. We have to calculate our residual functional variability up to the delta dimension according to our definition and lemma's. In Chapter 4, we will discuss the usefulness of residual functional variability both analytically and numerically.*

Note that effective dimension only provide partial information about the difficulty in approximating integrals. Small effective dimension does not necessary guarantee the effectiveness of QMC.

Chapter 4

Path Generation Methods

Consider the problem of pricing a European derivative security. We use $h(\mathbf{S}) = h(S_1, \ldots, S_d)$ to denote its payoff function at maturity T years, which depends on the asset prices $S_j := S_{t_j}$ at equally spaced times $t_j = j\Delta t$ for $j = 1, \ldots, d$ and $\Delta t = T/d$. Under the Black-Scholes (BS) model, the risk-neutral process of the underlying asset is given by 1.1. The analytical solution to (1.1) is given by

$$S_t = S_0 \exp\left(\left(r - \sigma^2/2\right) t + \sigma B_t\right). \tag{4.1}$$

According to the option pricing theory, the value of the financial derivative at $t = 0$ is $\mathbb{E}\left[e^{-rT} h(\mathbf{S})\right]$, where $\mathbb{E}\left[\cdot\right]$ is the expectation under the risk-neutral measure. For example, the price of a European arithmetic Asian option is $\mathbb{E}\left[e^{-rT} \max(S_A - K, 0)\right]$, where S_A is the arithmetic average of the underlying asset prices at times $t_1, \ldots, t_d$ and K is the strike price.

We now focus on pricing derivatives. Let $\boldsymbol{x} := (x_1, \ldots, x_d)^T \sim N_d(\mathbf{0}, \boldsymbol{\Sigma})$; i.e, $\boldsymbol{x}$ is normally distributed with mean $\mathbf{0}$ and covariance matrix $\boldsymbol{\Sigma}$ with its entry given by

$$\boldsymbol{\Sigma}_{ij} = \min(t_i, t_j) = \Delta t \, \min(i, j), \tag{4.2}$$

then using (4.1), the payoff function $g(\mathbf{S})$ can be expressed in term of $\boldsymbol{x}$ as

$$h(\mathbf{S}) = g(\exp(\mu_1 + \sigma x_1), \ldots, \exp(\mu_d + \sigma x_d)) := g(\boldsymbol{x}), \tag{4.3}$$

where $\mu_j = \log S_0 + (r - \sigma^2/2)\, t_j$. Note that we have redefined $h(\mathbf{S})$ as $g(\boldsymbol{x})$ to emphasize the explicit role of $\boldsymbol{x}$ in Chapter 1.

From the point of view of integration, by setting $\boldsymbol{x} = \boldsymbol{A}\,\boldsymbol{z}$, where $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{\Sigma}$, $\boldsymbol{z} = (z_1, \ldots, z_d)^T \sim N_d(\mathbf{0}, I)$ and I is the $d \times d$ identity matrix, and then imposing the transformation $\boldsymbol{z} = \Phi^{-1}(\boldsymbol{u}) = (\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_d))^T$, the change of variables $\boldsymbol{x} = \boldsymbol{A}\,\boldsymbol{z}$ is equivalent to the generation of the Brownian motions

$$(B_1, \ldots, B_d)^T = \boldsymbol{A}\,(z_1, \ldots, z_d)^T, \tag{4.4}$$

where B_j is the Brownian motion defined in (4.1). Hence we refer to $\boldsymbol{A}$ as the *generating matrix* of the Brownian motion. A key insight to the above transformation is that $\boldsymbol{A}$ can be arbitrary as long as it satisfies $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{\Sigma}$. Consequently, different specification of $\boldsymbol{A}$ gives rise to different way of generating the Brownian motions. We collectively call these methods as the path generating methods (PGMs) since they relate to the simulation of Brownian paths.

In particular, it is widely belived that techniques that reduce the effective dimension of the function G increases the efficiency of QMC. As such, many dimension reduction techniques such as the principal component analysis (PCA), orthogonal transformation (OT), fast orthogonal transformation (FOT), Brownian bridge (BB), linear transformation (LT) have been proposed. These methods are reviewed in the next section. Section 4.2 describes our newly proposed dimension reduction technique which we denote as the Directional Control (DC) method. We show that the DC method can recover many of the existing PGMs. We also demonstrate that maximizing the variability of f according to the delta effective dimension by our DC method is computationally efficient, as shown in Section 4.3 which is devoted to analyzing the time complexity of constructing the decomposition matrix by all of these PGMs.

4.1 Path Generation Method (PGM)

The objective of this section is to provide a brief overview of the existing PGMs, including the standard way of generating the Brownian motions.

4.1.1 Forward or Standard (STD) Construction

The standard construction generates the Brownian motion sequentially as follow: given $B_0 = 0$

$$B_{t_j} = B_{t_{j-1}} + \sqrt{t_j - t_{j-1}} z_j, z_j \sim N(0,1), j = 1, \ldots, n.$$

This method takes $O(n)$ operations to generate a path. The corresponding generating matrix $\boldsymbol{A}$ is the Cholesky decomposition of the covariance matrix $\boldsymbol{\Sigma}$, which takes the form

$$\boldsymbol{A} = \boldsymbol{A}^{STD} = \begin{pmatrix} \sqrt{t_1} & 0 & \cdots & 0 \\ \sqrt{t_1} & \sqrt{t_2 - t_1} & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \sqrt{t_1} & \sqrt{t_2 - t_1} & \cdots & \sqrt{t_n - t_{n-1}} \end{pmatrix}.$$

4.1.2 Principal Component Analysis (PCA)

Acworth et al. (1998) proposed the PCA construction, which is based on an eigenvalue decomposition of the covariance matrix $\boldsymbol{\Sigma}$, such that $\boldsymbol{\Sigma} = \boldsymbol{V}^T \boldsymbol{\Lambda} \boldsymbol{V}$, where $\boldsymbol{V}$ is the matrix of its eigenvectors and $\boldsymbol{\Lambda}$ is a diagonal matrix of its eigenvalues. This method maximises the concentration of the total variance of the Brownian path in the first few dimensions. The path is obtained as follows

$$\boldsymbol{A} = \boldsymbol{A}^{\boldsymbol{PCV}} = \sqrt{\boldsymbol{\Lambda}} \boldsymbol{V} = \begin{pmatrix} \sqrt{\lambda_1} v_{11} & \ldots & \sqrt{\lambda_d} v_{1d} \\ \vdots & \ddots & \vdots \\ \sqrt{\lambda_1} v_{d1} & \ldots & \sqrt{\lambda_d} v_{dd} \end{pmatrix},$$

where v_{ij} denotes the j^{th} coordinate of the i^{th} eigenvector and λ_i denotes the i^{th} eigenvalue of the covariance matrix $\mathbf{\Sigma}$ satisfying $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$. The eigenvectors and eigenvalues are given by

$$v_{ij} = \frac{2}{\sqrt{2d+1}} \sin\left(\frac{2i-1}{2d+1} j\pi\right) \tag{4.5}$$

$$\lambda_i = \frac{\Delta t}{4} \left(\sin\left(\frac{2i-1}{4d+2}\pi\right)\right)^{-2}. \tag{4.6}$$

Since $\boldsymbol{A}$ is a full matrix, the PCA construction requires $O\left(d^2\right)$ operations for the generation of one path instead of $O(d)$ operations which are needed by forward construction. For large d, this often increases the run times of the simulation and limits the practical use of the PCA construction. Akesson and Lehoczky (1998) showed that for $k = 1, ..., d$, the k-th eigenvalue and eigenvector of the matrix $\mathbf{\Sigma}$ can also be written as

$$\lambda_k = \left(4d \sin^2\left(\frac{2k-1}{2d+1}\right)\frac{\pi}{2}\right)^{-1}$$

and $\boldsymbol{v_k} = (v_{k,1}, ..., v_{k,d})^T$, where

$$v_{k,j} = \frac{2}{\sqrt{2d+1}} \sin\left(\frac{(2k-1)j}{2d+1}\pi\right), \text{ for } j = 1, ..., d,$$

respectively. Therefore $\mathbf{\Sigma} = \boldsymbol{V}\boldsymbol{D}(\boldsymbol{V}\boldsymbol{D})^T$, where $\boldsymbol{V}$ is the eigenvector matrix $\boldsymbol{V} = (\boldsymbol{v_1}$, $\cdots$, $\boldsymbol{v_d})$ and $\boldsymbol{D}$ is the diagonal matrix that has $\lambda_1^{1/2}, ..., \lambda_d^{1/2}$ as its diagonal elements. By using this construction, Scheicher (2007) has showed that PCA can be computed using the fast sine transform, which uses $O(d \log(d))$ flops.

4.1.3 Brownian Bridge (BB)

The BB method, which was first proposed by Moskowitz and Caflisch (1996), simulates the Brownian motion by first generating the final value B_d, and then samples the intermediate values $B_{\lfloor d/2 \rfloor}$ conditional on the values of B_d and B_0. After that, this method recursively

fills the intermediate values on $\left[B_0, B_{\lfloor d/2 \rfloor}\right)$ and $\left(B_{\lfloor d/2 \rfloor}, B_d\right)$, where $\lfloor x \rfloor$ denotes the greatest integer less than or equal to x. In particular, if d is a power of 2, then BB generates the Brownian motion as follows:

$$\begin{aligned} B_d &= \sqrt{T} Z_1, \\ B_{d/2} &= \frac{1}{2}(B_0 + B_d) + \sqrt{T/4} Z_2, \\ B_{d/4} &= \frac{1}{2}\left(B_0 + B_{d/2}\right) + \sqrt{T/8} Z_3, \\ &\cdots \\ B_{d-1} &= \frac{1}{2}\left(B_{d-2} - B_d\right) + \sqrt{T/2d} Z_d, \end{aligned} \tag{4.7}$$

where Z_i are independent standard normal random variables for $i = 1, ..., d$. BB construction corresponds to a certain matrix $\boldsymbol{A_{BB}}$ such that $\boldsymbol{A_{BB}} \boldsymbol{A_{BB}}^T = \boldsymbol{\Sigma}$. For example, if $T = 4$ and $\Delta t = 1$, then the decomposition matrix corresponds to the BB method is

$$\boldsymbol{A_{BB}} = \begin{pmatrix} 1/2 & 1/2 & \sqrt{2}/2 & 0 \\ 1 & 1 & 0 & 0 \\ 3/2 & 1/2 & 0 & \sqrt{2}/2 \\ 2 & 0 & 0 & 0 \end{pmatrix}.$$

More generally, suppose we are interested in constructing a discrete Brownian path $(B_{t_1}, ..., B_{t_d})$ with covariance matrix $\boldsymbol{\Sigma}$.

Algorithm 4.1.1. *Suppose the elements of $(B_{t_1}, ..., B_{t_d})$ should be computed in the order of $B_{t_{\pi(1)}}$, $B_{t_{\pi(2)}}$, ... , $B_{t_{\pi(d)}}$ for some permutation π of d Brownian paths. Consequently, in computing $B_{t_{\pi(j)}}$, we need to take into account the previously computed elements. Fortunately at most two of those are of relevance, the one next to $\pi(j)$ on the left and the one next to $\pi(j)$ on the right. Now, define for every $j \in \{1, ..., n\}$, $L(j) := \{k : k < \pi(j)$ and $\pi^{-1}(k) < j\}$, and $R(j) := \{k : k > \pi(j)$ and $\pi^{-1}(k) < j\}$. That is, L contains all the indices k that are smaller than $\pi(j)$ for which B_{t_k} has already been constructed and R*

contains all the indices k that are greater than $\pi(j)$ for which B_{t_k} has already been constructed. Then, if we let

$$l(j) := \begin{cases} 0 & if\ \ L(j) = \phi \\ \max L(j) & if\ \ L(j) \neq \phi \end{cases},$$

$$r(j) := \begin{cases} \infty & if\ \ R(j) = \phi \\ \min R(j) & if\ \ R(j) \neq \phi \end{cases},$$

and set $B_{t_0} = 0$, we have

$$B_{t_{\pi(j)}} := \begin{cases} B_{t_{l(j)}} + \sqrt{t_{\pi(j)} - t_{l(j)}} Z_j & if\ \ r(j) = \infty \\ \frac{t_{r(j)} - t_{\pi(j)}}{t_{r(j)} - t_{l(j)}} B_{t_{l(j)}} + \frac{t_{\pi(j)} - t_{l(j)}}{t_{r(j)} - t_{l(j)}} B_{t_{r(j)}} + \sqrt{\frac{(t_{\pi(j)} - t_{l(j)})(t_{r(j)} - t_{\pi(j)})}{t_{r(j)} - t_{l(j)}}} Z_j & if\ \ r(j) < \infty \end{cases}.$$

It is straightforward to check that the vector $B_{t_1}, ..., B_{t_n}$ constructed in this way has again covariance matrix $(\min(t_j, t_k))_{j,k}$. The functions l and r, as well as the factors of $B_{t_{l(j)}}, B_{t_{r(j)}}, Z_j$, do not depend on the random vector $\boldsymbol{Z}$ so their computation needs to be done only once. It is obvious that the Brownian bridge construction uses $O(n)$ floating point operations. In particular, the classical BB construction are constructed in the order B_1, $B_{1/2}$, $B_{1/4}$, $B_{3/4}$, $\cdots$.

The above description of the BB gives one implementation of generating the discrete Brownian motions. The optimal permutation of the Brownian bridge construction in the sense of explained variability is proved by Lin and Wang (2008) according to the following three theorems:

Theorem 4.1.1. *In a permutation-based construction of the Brownian motion $B_1, ..., B_d$, the optimal first step π_1 is the integer nearest to $(6d + 3)/8$.*

Theorem 4.1.2. *In a permutation-based construction of the Brownian motion $B_1, ..., B_d$, assume that B_q has been sampled (for some q with $0 \leq q \leq d - 1$), while $B_{q+1}, ..., B_d$ have not yet been generated. Among $B_{q+1}, ..., B_d$ conditional on the past value B_q, the local optimal new step q_{new} is the integer nearest to $(6d + 2q + 3)/8$.*

Theorem 4.1.3. *In a permutation-based construction of the Brownian motion $B_1, ..., B_d$, assume that B_{q_1} and B_{q2} have been sampled (for some q_1, q_2 with $0 \leq q_1 < q_2 \leq d$), while*

$B_{q_1+1}, ..., B_{q_2 1}$ have not yet been generated. Among $B_{q_1+1}, ..., B_{q_2 1}$ conditional on the past value B_{q1} and the future value B_{q2}, the optimal new step is the integer nearest to $(q_1+q_2)/2$.

The optimal permutations for $d = 2, 4, 8, 16, 32, 64, 128, 256$ are provided in Appendix A.

4.1.4 Orthogonal Transformation (OT)

Wang and Tan (2012) proposed an OT method for generating Brownian paths under Barrier options.

Theorem 4.1.4. *(Wang and Tan) Let $\boldsymbol{C}$ be a $d \times d$ positive definite matrix and let $\boldsymbol{A_0}$ be a fixed decomposition matrix such that $\boldsymbol{A_0 A_0}^T = \boldsymbol{C}$. Suppose that the indicator function $\Lambda(\boldsymbol{x})$ has the form*

$$\Lambda(\boldsymbol{x}) = I_{h(\boldsymbol{q}^T\boldsymbol{x})}(\boldsymbol{x}),\ \boldsymbol{x} = (x_1, ..., x_d)^T \sim N_d(\boldsymbol{0}, \boldsymbol{C}), \tag{4.8}$$

where $\boldsymbol{q} = (q_1, ..., q_d)^T$ is a vector of constants and $h(\cdot)$ is a function defined in R. If U is a $d \times d$ orthogonal matrix, whose first column $\boldsymbol{U_1}$ is given by

$$\boldsymbol{U_1} = \frac{1}{D}\boldsymbol{A_0}^T\boldsymbol{q}, \tag{4.9}$$

where $D := \sqrt{\boldsymbol{q}^T\boldsymbol{C}\boldsymbol{q}}$, the remaining columns are arbitrary as long as they satisfy the orthogonality conditions, then by the transformation $\boldsymbol{x} = \boldsymbol{A_0 U z}$, the function $h(\boldsymbol{q}^T\boldsymbol{x})$ involved in the indicator function $\Lambda(\boldsymbol{x})$ is transformed to a function depending only on the first component of $\boldsymbol{z}$:

$$h(\boldsymbol{q}^T\boldsymbol{x}) = h(Dz_1).$$

Consequently, the indicator function $\Lambda(\boldsymbol{x})$ is transformed to

$$\Lambda(\boldsymbol{x}) = I_{h(Dz_1)<H}(\boldsymbol{z}),\ \boldsymbol{z} = (z_1, ..., z_d)^T \sim N_d(\boldsymbol{0}, (I)).$$

If $h(\cdot)$ is strictly increasing on R and if $I_{h(Dz_1)<H}(\boldsymbol{z})$ is further transformed by the inverse normal transformation $\boldsymbol{z} = \Phi^{-1}(\boldsymbol{u})$, then the indicator function $\Lambda(\boldsymbol{x})$ is transformed to a one-dimensional function:

$$\Lambda(\boldsymbol{x}) = I_{u_1<c}(\boldsymbol{u}),\ \boldsymbol{u} = (u_1, ..., u_d)^T \sim U(0,1)^d,$$

where $c = \Phi(D^{-1}h^{-1}(H))$ *is a constant. The discontinuities of the indicator function* $I_{u_1<c}(\boldsymbol{u})$ *are aligned with the coordinate axes, which are QMC-friendly.*

Proof. See Appendix C. □

Theorem 4.1.4 offers a new PGM for simulating Brownian motion. Wang and Tan (2012) referred to this new PGM as the orthogonal transformation (OT). The OT method is simple to implement by constructing an orthogonal matrix $\boldsymbol{U}$ according to Theorem 4.1.4 and taking the generating matrix to be $\boldsymbol{A} = \boldsymbol{A_0}\boldsymbol{U}$ for some fixed $\boldsymbol{A_0}$ satisfying $\boldsymbol{A_0}\boldsymbol{A_0}^T = \boldsymbol{C}$. Whenever a function involves an indicator function of the form (4.8), Theorem 4.1.4 guarantees that the discontinuities are aligned with the coordinate axes. When a function involves an indicator function which is not exactly the form (4.8), but is "close"to this form in some sense, Theorem 4.1.4 is still useful in finding a good PGM as illustrated by Wang and Tan (2012). The value of the simple situation in Theorem 4.1.4 is to give insight of a function.

From a practical point of view, Wang and Tan (2012) mentioned two issues in Theorem 4.1.4.

1. Theorem 4.1.4 gives us only the first column of $\boldsymbol{U}$. Other columns of $\boldsymbol{U}$ are found by the Gram-Schmidt method or modified Gram-Schmidt algorithm (Wang and Sloan 2010). The determination of other columns also leaves room for further optimization of the generating matrix by taking into account the knowledge of the function.

2. Another issue is the choice of the initial decomposition matrix $\boldsymbol{A_0}$. If the underlying integrand is, say, a product of an indicator function of the form (4.8) with another function $G_0(\boldsymbol{x})$, i.e. $G(\boldsymbol{x}) = G_0(\boldsymbol{x})\Lambda(\boldsymbol{x})$, then the choice of $\boldsymbol{A_0}$ could have impact on the practical performance of QMC methods, since Theorem 4.1.4 only focuses on the indicator function $\Lambda(\boldsymbol{x})$. If there is an indication that the function $G_0(\boldsymbol{x})$ is PCA-friendly, then we may choose the initial decomposition matrix $\boldsymbol{A_0}$ to be $\boldsymbol{A_{PCA}}$. If no priori information is available, then $\boldsymbol{A_0}$ is taken to be the Cholesky decomposition of $\boldsymbol{C}$.

4.1.5 Fast Orthogonal Transformation (FOT)

Leobacher (2012) proposed some FOT methods for generating Brownian paths.

Definition 4.1.1. *A linear transformation T from R^n to R^n is called orthogonal if it preserves the length of vectors:*

$$\|T(\boldsymbol{x})\| = \|\boldsymbol{x}\|, \text{ for all vector } \boldsymbol{x} \text{ in } R^n.$$

If $T(\boldsymbol{x}) = \boldsymbol{T}\boldsymbol{x}$ is an orthogonal transformation, we say that $\boldsymbol{T}$ is an orthogonal matrix.

Then, according to Theorem 4.1.5 below, we conclude that any decomposition matrix can be constructed by the FOT method with an initial decomposition matrix. In other words, with properly choosing an orthogonal transformation matrix, the FOT method can recover all decomposition matrices.

Theorem 4.1.5. *(Papageorgiou 2002)*

1. *If $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^T$ is the Cholesky decomposition of $\boldsymbol{\Sigma}$, then any orthogonal transformation T on R^n defines a decomposition $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{T}(\boldsymbol{L}\boldsymbol{T})^T$.*
2. *Conversely, for every n by n matrix $\boldsymbol{A}$ with $\boldsymbol{\Sigma} = \boldsymbol{A}\boldsymbol{A}^T$, there exists an orthogonal transform $\boldsymbol{T}$ such that $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{T}$.*

Proof: For any orthogonal transform $\boldsymbol{T}$, we have

$$\boldsymbol{T}^T = \boldsymbol{T}^{-1},$$

such that $\boldsymbol{L}\boldsymbol{T}(\boldsymbol{L}\boldsymbol{T})^T = \boldsymbol{L}\boldsymbol{T}\left(\boldsymbol{T}^T\boldsymbol{L}^T\right) = \boldsymbol{L}\boldsymbol{L}^T$. Thus, statement (1) is proved.
On the other hand, $\boldsymbol{L}$ is invertible, so that for $\boldsymbol{T} = \boldsymbol{L}^{-1}\boldsymbol{A}$ we have, $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{T}$, and

$$\boldsymbol{T}\boldsymbol{T}^T = \boldsymbol{L}^{-1}\boldsymbol{A}\boldsymbol{A}^T\left(\boldsymbol{L}^{-1}\right)^T = \boldsymbol{L}^{-1}\boldsymbol{\Sigma}\left(\boldsymbol{L}^{-1}\right)^T = \boldsymbol{L}^{-1}\boldsymbol{L}\boldsymbol{L}^T\left(\boldsymbol{L}^T\right)^{-1} = \boldsymbol{I}$$

so that $\boldsymbol{T}$ is orthogonal. Thus statement (2) is proved. □

The purpose of using orthogonal transformation is to get the benefits of fast matrix-vector multiplication. Probably the most famous example of a unitary transform that allows for fast matrix-vector multiplication is the discrete Fourier transform DFT (with the corresponding fast multiplication algorithm: fast Fourier transform(FFT)). There are many

variants of the discrete Fourier transform that map real functions to real functions, where the transformation matrix is an orthogonal matrix. We present some of those famous ones.

Discrete Fourier Transform

An N-point DFT is expressed as an N-by-N matrix multiplication as $\boldsymbol{X} = \boldsymbol{F}\boldsymbol{x}$, where x is the original input signal, and $\boldsymbol{F}$ is the DFT of the signal.
The transformation $\boldsymbol{F}$ of size $N*N$ can be defined as $\boldsymbol{F} = \left(\frac{\omega^{jk}}{\sqrt{N}}\right)_{j,k=0,\ldots,N-1}$, or equivalently:

$$F = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(N-1)} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{pmatrix}.$$

where $\omega = e^{-\frac{2\pi i}{N}}$ is a primitive Nth root of unity in which $i = \sqrt{-1}$. Then, it's easy to check that $\boldsymbol{F}$ is an orthogonal matrix.

Modified Fourier Transform

Most Fourier variants that map real functions to real functions have some additional special properties which, for example, make them useful for the fast generation of convolutions. From our perspective, we want to have many different variants to choose from for a particular application. Leobacher (2012) presented one modification of the Fourier transform that maps real functions to real functions. It is well known that the discrete Fourier transform of a vector $\boldsymbol{x} = (x_0, ..., x_{n-1})$ is real-valued iff $x_{n-k} = \bar{x_k}$. Consider the linear map $\boldsymbol{B}$,

$$(\boldsymbol{Bx})_k = \begin{cases} x_0, & if\, k = 0 \\ x_k + ix_{\lfloor n/2 \rfloor + k}, & if\, 0 < k < n/2 \\ x_{n/2}, & if\, k = n/2 \\ x_{\lfloor n/2 \rfloor - k} - ik_{n-k}, & if\, n/2 < k < n \end{cases}.$$

Then for any standard normal vector $\boldsymbol{X}$, $\boldsymbol{BX}$ is also a (real-valued) standard normal vector.

There is a drawback of this method since the essential of FFT algorithm uses complex multiplication, which uses 4 real multiplications. So finding some fast orthogonal transforms that use only real multiplications is important to improve the construction time complexity.

Sine and Cosine Transform (DCT)

Two of the most important orthogonal transforms that only use real operations are the sine and cosine transform. There are four widely used variants of the sine and cosine transform. For the definitions we refer to the Wang (1984) as follow:

Discrete Sine Transform:

$$C(\boldsymbol{x}) := \left(\sqrt{\tfrac{2}{n}} \sum_{k=1}^{n} x_k \sin\left(\tfrac{\pi}{n} \left(k - \tfrac{1}{2} \right) \left(j - \tfrac{1}{2} \right) \right) \right)_{j=1}^{n}.$$

Discrete Cosine Transform:

$$C(\boldsymbol{x}) := \left(\sqrt{\tfrac{2}{n}} \sum_{k=1}^{n} x_k \cos\left(\tfrac{\pi}{n} \left(k - \tfrac{1}{2} \right) \left(j - \tfrac{1}{2} \right) \right) \right)_{j=1}^{n}.$$

Now, we only look at the discrete cosine transform, and we are going to show that $\boldsymbol{LC}$ is approximating to the eigenvalue decomposition, where $\boldsymbol{C}$ is the DCT matrix, and $\boldsymbol{L}$ is

defined as

$$\boldsymbol{L} = \frac{1}{\sqrt{n}} \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix}.$$

Proof. See Appendix B. □

Hence, the Cosine Transform construction is close to the eigenvalue decomposition. The numerical results will be shown in Chapter 5. However, there's no clue of the optimal criterion for the orthogonal transformation with different payoff functions, so the orthogonal transformation according to the FFT algorithm may not be helpful for the dimension reduction in a particular payoff function.

4.1.6 Linear Transformation

To derive the optimal orthogonal matrix $\boldsymbol{A}$ in general functions, Imai and Tan (2006) proposed to approximate the objective function by linearizing the function g and then maximizing the variance contribution according the first-order Taylor expansion to a general function g at around an arbitrary point $\boldsymbol{\epsilon} = \hat{\boldsymbol{\epsilon}} + \Delta\boldsymbol{\epsilon}$:

$$g(\boldsymbol{\epsilon}) \approx g\left(\hat{\boldsymbol{\epsilon}}\right) + \sum_{k=1}^{d} \frac{\partial g}{\partial \epsilon_k}|_{\boldsymbol{\epsilon}=\hat{\boldsymbol{\epsilon}}} \Delta\epsilon_k, \tag{4.10}$$

and the approximated function is linear in the normal random variables $\Delta\boldsymbol{\epsilon}$ with

$$\left(\frac{\partial g}{\partial \epsilon}_k|_{\boldsymbol{\epsilon}=\hat{\boldsymbol{\epsilon}}}\right)^2 \tag{4.11}$$

to be the variability contributed by the k^{th} component. The optimization of $\boldsymbol{A}$ is given by

$$\max_{\boldsymbol{A}_{,\boldsymbol{k}} \in R^d} \left(\frac{\partial g}{\partial \epsilon_k}|_{\boldsymbol{\epsilon}=\hat{\boldsymbol{\epsilon}}}\right)^2$$

subject to $\|\boldsymbol{A}_{\boldsymbol{.k}}\|$ and $\langle \boldsymbol{A}^{*}_{\boldsymbol{.j}}, \boldsymbol{A}_{\boldsymbol{.k}} \rangle = 0$, for $j = 1, ..., k-1$.

The optimal solution turns out to be very simple when considering the following theorem.

Theorem 4.1.6. *Let* $\boldsymbol{V} = span\{\boldsymbol{q_1}, ..., \boldsymbol{q_{k-1}}\}$, $\boldsymbol{\sigma}_k = \left(\frac{\partial g}{\partial \epsilon_k}(\boldsymbol{a})\right)^2$, *and* $\boldsymbol{w} = \nabla g(\boldsymbol{Qa})$ *for* $\boldsymbol{a} = (1, ..., 1, 0, ..., 0)$ *with* $k-1$ *leading ones in the* k^{th} *step. Assume that* $\boldsymbol{w} \notin \boldsymbol{V}$, *then*

$$\boldsymbol{q_k} = \pm \overline{\boldsymbol{w}} / \|\overline{\boldsymbol{w}}\| \tag{4.12}$$

with $\overline{\boldsymbol{w}} := \boldsymbol{w} - \sum_{i=1}^{k-1} \langle \boldsymbol{w}, \boldsymbol{q_k} \rangle \boldsymbol{q_k}$ *is the unique solutions of the constrained optimization problem*

$$\max_{\boldsymbol{q_k} \in \overline{\boldsymbol{V}}} \boldsymbol{\sigma}_k \text{ with } \|\boldsymbol{q_k}\| = \boldsymbol{1}, \tag{4.13}$$

where $\overline{\boldsymbol{V}} \subset R^d$ *denotes the orthogonal complement space of* $\boldsymbol{V}$.

4.2 Directional Control (DC) Method: The Proposed Method

4.2.1 Introduction

Recall that our objective is to provide an effective QMC-based method for simulating derivative securities with payoff at maturity given by

$$g(\boldsymbol{x}) = \max(f(\boldsymbol{x}), 0),$$

where f is a function that depends on d correlated multivariate normal $\boldsymbol{x} = (x_1, ..., x_d)^T \sim N_d(\boldsymbol{0}, \boldsymbol{\Sigma})$. This boils down to seeking a decomposition matrix $\boldsymbol{A}$ satisfying $\boldsymbol{AA}^T = \boldsymbol{\Sigma}$. Let us now define a vector of random variables $\boldsymbol{Y} = \{f_1(x_1, ..., x_d), \ldots, f_d(x_1, ..., x_d)\}$ such that there exists a function ψ such that

$$f(\boldsymbol{x}) = \psi(f_1, ..., f_d). \tag{4.14}$$

As an example, let us consider pricing an Asian option under the classical Black-Scholes model. In this case, we have $f_i(\boldsymbol{x}) = S_0 e^{(r-\sigma^2/2)t_i+\sigma x_i}$ so that f_i represents the stock price at time t_i. More importantly f_i depends only on a single random variable x_i and that (4.14) becomes an additive function of the form

$$f(\boldsymbol{x}) = f_1(x_1) + \cdots + f_d(x_d).$$

In general, let us apply the first-order vector Taylor expansion to each of the function f_i, for $i = 1, ..., d$, around an arbitrary vector $\boldsymbol{x} = \hat{\boldsymbol{x}} + \Delta\boldsymbol{x}$:

$$\boldsymbol{Y} \approx \boldsymbol{Y}(\hat{\boldsymbol{x}}) + \nabla\boldsymbol{Y}(\hat{\boldsymbol{x}})(\boldsymbol{x} - \hat{\boldsymbol{x}}). \tag{4.15}$$

Using the Delta method, the covariance matrix of $\boldsymbol{Y}$ is defined as follows:

$$\boldsymbol{\Sigma}_f = Var(\boldsymbol{Y}) = \boldsymbol{J}\boldsymbol{\Sigma}\boldsymbol{J}^T \tag{4.16}$$

where $\boldsymbol{J}$ is the Jacobian matrix of $\boldsymbol{Y}$ with respect to $\boldsymbol{x}$; i.e.

$$\begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_d} \\ \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_2}{\partial x_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_d}{\partial x_1} & \cdots & \frac{\partial f_d}{\partial x_d} \end{pmatrix}.$$

The key insight to (4.16) is that if we have a decomposition matrix $\boldsymbol{A}_f$ satisfying $\boldsymbol{A}_f\boldsymbol{A}_f^T = \boldsymbol{\Sigma}_f$, then the corresponding decomposition matrix of $\boldsymbol{\Sigma}$ can be obtained via

$$\boldsymbol{A} = \boldsymbol{J}^{-1}\boldsymbol{A}_f. \tag{4.17}$$

Hence this provides another approach of determining $\boldsymbol{A}$ indirectly via $\boldsymbol{A}_f$. In contrast to the direct approach of determining $\boldsymbol{A}$, the proposed indirect approach depends explicitly on f. For example, as we have seen that under the PCA the decomposition matrix $\boldsymbol{A}$ is obtained by maximizing the explained variability of $\|\boldsymbol{x}\|$. If the same PCA approach is used in our proposed indirect approach, $\boldsymbol{A}_f$, which maximizes the explained variability of

$\|\boldsymbol{Y}\|$, is first obtained and (4.17) is then used to derive the desired decomposition matrix $\boldsymbol{A}$. The latter approach is more appropriate, especially if $\boldsymbol{Y}$ is a non-linear function of $\boldsymbol{x}$, since ultimately we are concerned with simulating f. The above discussion also implies that when f_i is a linear function of x_i and ψ is an additive function, then applying PCA to both methods yields exactly the same decomposition matrix $\boldsymbol{A}$.

Remark 4.2.1. *If rank($\boldsymbol{J}$) $< d$, Moore (1920) showed that the inverse of $\boldsymbol{J}$ is not unique. Here, we only consider the case where the rank of functional random variables equals to d; i.e. there does not exist any random variable f_i which is a linear combination of the others.*

In the next subsection, we delve into the issue of optimal selection of $\boldsymbol{A}_f$. In particular, we propose three directional control methods for finding the optimal decomposition matrix A_f. We refer to these methods as projection control (PC), sequential control (SC), and mixture control (MIXC). Intuitively, the methods of PC, SC, and MIXC can be understood as controlling, respectively, the column directions, row directions, and both column and row directions.

4.2.2 Projection Control (PC) Method

The purpose of projection control is to construct the columns of the decomposition matrix of $\boldsymbol{\Sigma}$ according to controlling the column directions of the functional decomposition matrix of $\boldsymbol{\Sigma}_f$. We first discuss the idea of projection matrix, also called "hat matrix", and then we introduce our proposed PC method.

Hat Matrix

For a n by m matrix $\boldsymbol{X}$, the square matrix $\boldsymbol{H}$ of dimension n defined as $\boldsymbol{H} = \boldsymbol{X}\ (\boldsymbol{X}^T\boldsymbol{X})^{-1}\ \boldsymbol{X}^T$ is known as the "hat matrix"or the "projection matrix". The matrix $\boldsymbol{H}$ has the following properties:

1. idempotent: $\boldsymbol{H}\boldsymbol{H} = \boldsymbol{H}$, and symmetric: $\boldsymbol{H}^T = \boldsymbol{H}$.
2. $\boldsymbol{H}$ is positive semi-definite. i.e. $\boldsymbol{v}^T\boldsymbol{H}\boldsymbol{v} \geq 0$, for any $\boldsymbol{v}$ in R^n.
3. All eigenvalues of $\boldsymbol{H}$ are either 0 or 1 with p eigenvalues equal to 1 and $n-p$ eigenvalues equal to 0, where $p = dim(span\{\boldsymbol{X}\})$.
4. $trace(\boldsymbol{H}) = \sum_{i=1}^{n} h_{i,i} = p$.

Hence, $\boldsymbol{H}$ is a projection matrix onto a p-dim subspace.

Orthogonal Hat Matrix

The orthogonal matrix $\boldsymbol{I} - \boldsymbol{H}$ is also a "Hat Matrix"with the following properties:

1. $\boldsymbol{I} - \boldsymbol{H}$ is idempotent and symmetric.
2. $\boldsymbol{I} - \boldsymbol{H}$ is positive semi-definite.
3. All eigenvalues are either 0 or 1 with $n-p$ eigenvalues equal to 1 and p eigenvalues equal to 0.
4. $trace(\boldsymbol{I} - \boldsymbol{H}) = n - p$.
5. $\boldsymbol{H}(\boldsymbol{I} - \boldsymbol{H}) = \boldsymbol{H} - \boldsymbol{H}\boldsymbol{H} = \boldsymbol{H} - \boldsymbol{H} = \boldsymbol{0}$. i.e. $\boldsymbol{H}$ and $\boldsymbol{I} - \boldsymbol{H}$ are orthogonal.

Hence, $\boldsymbol{I} - \boldsymbol{H}$ is a project matrix onto orthogonal complement space of $span\{\boldsymbol{X}\}$.

Projection Control

For any d by m direction matrix $\boldsymbol{V}$, with $d = dim(\boldsymbol{\Sigma})$, and $\boldsymbol{A_0}$ being an arbitrary decomposition of $\boldsymbol{\Sigma}_f$, we have

$$\boldsymbol{H} = \boldsymbol{V}(\boldsymbol{V}^T\boldsymbol{V})^{-1}\boldsymbol{V}^T.$$

Then, we can construct

$$\boldsymbol{\Sigma}_f = \boldsymbol{A_0}\boldsymbol{H}\boldsymbol{A_0}^T + \boldsymbol{A_0}(\boldsymbol{I} - \boldsymbol{H})\boldsymbol{A_0}^T, \tag{4.18}$$

where $\boldsymbol{H}$ and $\boldsymbol{I}-\boldsymbol{H}$ are both projection matrices which project onto $span\{V\}$ and its orthogonal space respectively. In particular, let $\boldsymbol{V}=[\boldsymbol{v_1},\boldsymbol{v_2},...,\boldsymbol{v_d}]$ be our direction matrix, where any vector is orthogonal to the others. Then, for the first direction vector $\boldsymbol{v_1}$, we define $\boldsymbol{H_1}=\boldsymbol{v_1}(\boldsymbol{v_1}^T\boldsymbol{v_1})^{-1}\boldsymbol{v_1}^T$, and we have

$$\boldsymbol{\Sigma}_f=\boldsymbol{A_0}\boldsymbol{H_1}\boldsymbol{A_0}^T+\boldsymbol{A_0}(\boldsymbol{I}-\boldsymbol{H_1})\boldsymbol{A_0}^T.$$

Then, the second direction vector $\boldsymbol{v_2}$ is on the orthogonal space $\boldsymbol{I}-\boldsymbol{H_1}$. We define $\boldsymbol{H_2}=\boldsymbol{v_2}(\boldsymbol{v_2}^T\boldsymbol{v_2})^{-1}\boldsymbol{v_2}^T$, and we have

$$\boldsymbol{\Sigma}_f=\boldsymbol{A_0}\boldsymbol{H_1}\boldsymbol{A_0}^T++\boldsymbol{A_0}\boldsymbol{H_2}\boldsymbol{A_0}^T+\boldsymbol{A_0}(\boldsymbol{I}-\boldsymbol{H_1}-\boldsymbol{H_2})\boldsymbol{A_0}^T.$$

Similarly, for the direction $\boldsymbol{v_i}$ on the orthogonal space $\boldsymbol{I}-\sum_{j=1}^{i-1}\boldsymbol{H_j}$, we recursively define $\boldsymbol{H_i}=\boldsymbol{v_i}(\boldsymbol{v_i}^T\boldsymbol{v_i})^{-1}\boldsymbol{v_i}^T$, for $i=3,...,d$. Finally, we get

$$\boldsymbol{\Sigma}_f=\boldsymbol{A_0}\boldsymbol{H_1}\boldsymbol{A_0}^T+...+\boldsymbol{A_0}\boldsymbol{H_d}\boldsymbol{A_0}^T,$$

where $\boldsymbol{H_d}=\boldsymbol{I}-\boldsymbol{H_1}-\boldsymbol{H_2}-...-\boldsymbol{H_{d-1}}$ is the projection onto 1-dimensional space. Then the decomposition matrix of $\boldsymbol{\Sigma}_f$ is defined as

$$\boldsymbol{A_f}=[\boldsymbol{v_1'},...,\boldsymbol{v_d'}]=\left[\frac{\boldsymbol{A_0}\boldsymbol{v_1}}{\sqrt{\boldsymbol{v_1}^T\boldsymbol{v_1}}},...,\frac{\boldsymbol{A_0}\boldsymbol{v_d}}{\sqrt{\boldsymbol{v_d}^T\boldsymbol{v_d}}}\right], \tag{4.19}$$

and

$$\boldsymbol{A}=\boldsymbol{J}^{-1}\boldsymbol{A_f} \tag{4.20}$$

is a new decomposition of covariance matrix $\boldsymbol{\Sigma}$.

Now, we propose the following two theorems:

Theorem 4.2.1. *If $rank(\boldsymbol{\Sigma}_f)=d$, then there exists a projection decomposition such that our new decomposition becomes the same form as the eigenvalue decomposition of $\boldsymbol{\Sigma}$.*

Proof: Let $\boldsymbol{\Sigma}=\boldsymbol{V}^T\boldsymbol{\Lambda}\boldsymbol{V}$, where $\boldsymbol{V}$ is the matrix of its eigenvectors and $\boldsymbol{\Lambda}$ is a diagonal matrix of its eigenvalues. Note that $\boldsymbol{V}\boldsymbol{V}^T$ is the identity matrix; i.e. $\boldsymbol{V}$ is an orthogonal matrix. Then we have

$$\boldsymbol{\Sigma}_f \;=\; \boldsymbol{J}\boldsymbol{\Sigma}\boldsymbol{J}^T=\boldsymbol{J}\boldsymbol{V}\boldsymbol{\Lambda}^{\boldsymbol{1/2}}\boldsymbol{\Lambda}^{\boldsymbol{1/2}}\boldsymbol{V}\boldsymbol{J}^T.$$

Since for any arbitrary decomposition of $\boldsymbol{\Sigma}_f$, denoted as $\boldsymbol{A_0}$, there exists an orthogonal matrix $\boldsymbol{X} = (\boldsymbol{X_{.1}}, \boldsymbol{X_{.2}}, ..., \boldsymbol{X_{.d}})$ such that

$$\boldsymbol{A_0 X X}^T \boldsymbol{A_0}^T = \boldsymbol{\Sigma}_f, \tag{4.21}$$

and hence, our projection matrix could be constructed as

$$\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T = \boldsymbol{X}\boldsymbol{X}^T = \boldsymbol{I}.$$

Since $\boldsymbol{X}$ is an full rank orthogonal matrix, our projection matrix $\boldsymbol{H}$ is equivalent to the identity matrix $\boldsymbol{I}$. In this case, $\boldsymbol{X} = \boldsymbol{A_0}^{-1}\boldsymbol{JV\Lambda^{1/2}}$, and

$$\boldsymbol{\Sigma}_f = \boldsymbol{A_0} H \boldsymbol{A_0}^T,$$

which means that the projection control of the covariance matrix $\boldsymbol{\Sigma}_f$ according to the column vectors of matrix $\boldsymbol{X}$ is equivalent to the eigenvalue decomposition of $\boldsymbol{\Sigma}$. i.e. There exists a projection decomposition such that our new decomposition is equivalent to the eigenvalue decomposition. □

Moreover since the direction matrix does not care about the scale, so for any constants $c_1, \ldots, c_d$, $\boldsymbol{X} = (c_1\boldsymbol{X}_{\cdot 1}, c_2\boldsymbol{X}_{\cdot 2}, \ldots, c_d\boldsymbol{X}_{\cdot d})$ will have the same projection effect as $\boldsymbol{X}$.

Theorem 4.2.2. *If $rank(\boldsymbol{\Sigma}_f) = d$, for any decomposition matrix $\boldsymbol{A}$ of covariance matrix $\boldsymbol{\Sigma}$, there exists a a projection decomposition which is equivalent to $\boldsymbol{A}$.*

Proof: Recall from the Theorem 4.1.5 (Papageorgiou 2002): for any decomposition matrix $\boldsymbol{A}$ such that $\boldsymbol{\Sigma} = \boldsymbol{A}\boldsymbol{A}^T$, there exists an orthogonal transform $\boldsymbol{T}$ such that $\boldsymbol{A} = \boldsymbol{A_0 T}$ is a decomposition of $\boldsymbol{\Sigma}$, where $\boldsymbol{A_0}$ is an arbitrary decomposition of $\boldsymbol{\Sigma}$. Then, we can extend this result to the functional covariance matrix which states that for any decomposition matrix $\boldsymbol{A_f}$ such that $\boldsymbol{\Sigma}_f = \boldsymbol{A_f}\boldsymbol{A_f}^T$, there exists an orthogonal transform $\boldsymbol{T}$ such that $\boldsymbol{A_f} = \boldsymbol{A_{f_0}T}$ is a decomposition of $\boldsymbol{\Sigma}_f$, where $\boldsymbol{A_{f_0}}$ is an arbitrary decomposition of $\boldsymbol{\Sigma}_f$. Therefore, for any decomposition matrix $\boldsymbol{A}$, there exists an orthogonal transform $\boldsymbol{T}$ such

that

$$\begin{aligned} \boldsymbol{J}\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{J}^T &= \boldsymbol{\Sigma}_f \\ &= \boldsymbol{A}_{\boldsymbol{f_0}}\boldsymbol{T}\boldsymbol{T}^T{\boldsymbol{A}_{\boldsymbol{f_0}}}^T \\ &= \boldsymbol{A}_{\boldsymbol{f_0}}H{\boldsymbol{A}_{\boldsymbol{f_0}}}^T, \end{aligned}$$

where $\boldsymbol{H} = \boldsymbol{T}\boldsymbol{T}^T = \boldsymbol{T}(\boldsymbol{T}^T\boldsymbol{T})^{-1}\boldsymbol{T}^T = \boldsymbol{I}$, and $\boldsymbol{T} = {\boldsymbol{A}_{\boldsymbol{f_0}}}^{-1}\boldsymbol{J}\boldsymbol{A}$. So if we construct our project direction to be exactly the same as the directions of orthogonal transform $\boldsymbol{T}$, we will get a new decomposition $\boldsymbol{A}_{\boldsymbol{proj}} = \boldsymbol{A}_{\boldsymbol{f_0}}\boldsymbol{T} = \boldsymbol{J}\boldsymbol{A}$ such that $\boldsymbol{\Sigma}_f = \boldsymbol{A}_{\boldsymbol{proj}}{\boldsymbol{A}_{\boldsymbol{proj}}}^T$. Hence our decomposition matrix $\boldsymbol{A}_{\boldsymbol{x}}$ of the covariance matrix $\boldsymbol{\Sigma}$ according to our projection control method becomes $\boldsymbol{A}_{\boldsymbol{x}} = \boldsymbol{J}^{-1}\boldsymbol{A}_{\boldsymbol{proj}} = \boldsymbol{A}$. □

The above theorems imply that with a proper choice of direction matrix, PC method can recover any method (such as PCA and BB) of determining $\boldsymbol{A}$ directly from $\boldsymbol{\Sigma}$. Our proposed method, however, has the additional advantage of constructing a new decomposition matrix that tailors to specific directions while reflects certain desirable properties of f.

The optimal PC solution which maximizes the explained variability of $\|f\|$ (as constructed according to the PCA decomposition of $\boldsymbol{\Sigma}_f$), can be constructed from the algorithm below:

Algorithm 4.2.1. *Goal: Obtain the PCA decomposition matrix $\boldsymbol{A}_{f_{pca}}$ of $\boldsymbol{\Sigma}_f$, and then return the decomposition matrix as $\boldsymbol{A} = \boldsymbol{J}^{-1}\boldsymbol{A}_{f_{pca}}$.*

Step 1: $s = 1$

Step 2: For $k = s : d$

$$\boldsymbol{a_k} = \arg\max_{\boldsymbol{a_i}} {\boldsymbol{a_i}}^T \left(\boldsymbol{J}\left(\boldsymbol{\Sigma} - \hat{\boldsymbol{A}}_{\boldsymbol{k}}{\hat{\boldsymbol{A}}_{\boldsymbol{k}}}^T\right)\boldsymbol{J}^T\right)\boldsymbol{a_i}$$

(i.e. $\boldsymbol{a_k}$ is the k^{th} principal component of $\boldsymbol{J}\boldsymbol{\Sigma}\boldsymbol{J}^T$)

$$\boldsymbol{A}_{\boldsymbol{.k}} = \boldsymbol{J}^{-1}\boldsymbol{a_k}$$

End For

Step 3: Return $\boldsymbol{A} = [\boldsymbol{A}_{\boldsymbol{.1}}, \boldsymbol{A}_{\boldsymbol{.2}}, ..., \boldsymbol{A}_{\boldsymbol{.d}}]$.

Remark 4.2.2. *In Algorithm 4.2.1, k starting from an arbitrary number $s \leq d$ instead of* 1 *is similar to the singular value decomposition (SVD), which gives an optimal solution to maximize the functional variability for remaining $d - s + 1$ dimensions.*

Algorithm 4.2.2. *Goal: Suppose that we know some pre-determined controlling directions $\{\boldsymbol{v_1}, ..., \boldsymbol{v_{d_f}}\}$, then we use our projection matrix to construct the decomposition up to the dimension d_f and apply the SVD of $\boldsymbol{J}(\boldsymbol{\Sigma} - \hat{\boldsymbol{A}}_{d_f}\hat{\boldsymbol{A}}_{d_f}^T)\boldsymbol{J}^T$ for the remaining decomposition of $\boldsymbol{\Sigma}_f$.*

Step 1: $\hat{\boldsymbol{A}}_{\boldsymbol{d_f}f} = [\boldsymbol{v_1'}, ..., \boldsymbol{v_{d_f}'}] = \left[\frac{\boldsymbol{A_0 v_1}}{\sqrt{\boldsymbol{v_1}^T \boldsymbol{v_1}}}, ..., \frac{\boldsymbol{A_0 v_{d_f}}}{\sqrt{\boldsymbol{v_{d_f}}^T \boldsymbol{v_{d_f}}}}\right]$.

Step 2: $\hat{\boldsymbol{A}}_{\boldsymbol{d_f}} = \boldsymbol{J}^{-1}\hat{\boldsymbol{A}}_{\boldsymbol{d_f}f}$

Step 3: Get $\hat{\boldsymbol{A}}_{f_{remaining}}$ *by the SVD of* $\boldsymbol{J}(\boldsymbol{\Sigma} - \hat{\boldsymbol{A}}_{d_f}\hat{\boldsymbol{A}}_{d_f}^T)\boldsymbol{J}^T$. *(i.e. by Algorithm 4.2.1 with* $s = d_f + 1$*)*

Step 4: $\hat{\boldsymbol{A}}_{remaining} = \boldsymbol{J}^{-1}\hat{\boldsymbol{A}}_{f_{remaining}}$

Step 5: Return $\hat{\boldsymbol{A}} = \left[\hat{\boldsymbol{A}}_{d_f}; \hat{\boldsymbol{A}}_{remaining}\right]$

4.2.3 Sequential Control (SC) Method

The purpose of sequential control (SC) is to construct the rows of the decomposition matrix of $\boldsymbol{\Sigma}$ according to controlling the order of Brownian paths generation corresponding to the importance index function of the decomposed functional random variables f_i's, for $i = 1, ..., d$. Suppose that $\boldsymbol{x} = (x_1, ..., x_d)^T \in N_d(\boldsymbol{0}, \boldsymbol{\Sigma})$, and $\boldsymbol{x_\pi} = (x_{\pi_1}, ..., x_{\pi_d})$ such that $\{\pi_1, ..., \pi_d\}$ are the order of path generation. In other words, π_1 means that the Brownian path at time t_{π_1} is generated first, π_2 means that the Brownian path at time t_{π_2} is generated next, and so on up to the path at time t_{π_d} which is generated last. While using the idea of Brownian bridge, the discrete Brownian path can be generated in an arbitrary order, here we assume that the order of generation is defined by an importance index function of f_i's

as,

$$\boldsymbol{\pi} = \eta(f_1, ..., f_d), \tag{4.22}$$

where f_i's are functional random variables of x_i's defined in Section 4.2.1, and $\boldsymbol{\pi}$, which we denote as the importance index array, is a vector of the permutation of $\{1, ..., d\}$ and define the order of generating Brownian paths x_i's, for $i = 1, ..., d$.

The importance index function η is decided subjectively according to the specific payoff function. For example, in lookback options, the importance index array can be chosen to be the order of $\{E(f_1(x_1)), ..., E(f_d(x_d))\}$, where f_i is defined as the stock price at time t_i for $i = 1, ..., d$, with decreasing or increasing order depending on the payoff $\max(f_1(x_1), ..., f_d(x_d))$ or $\min(f_1(x_1), ..., f_d(x_d))$. i.e. In this case, η is defined as the expectation of the functional random variables. In the weighted average options, the importance index function can be chosen to be the order of $\{Var(f_1(x_1)), ..., Var(f_d(x_d))\}$ with decreasing or increasing order depending on the payoff $\max(f(\boldsymbol{x}) - K, 0)$ or $\min(f(\boldsymbol{x}) - K, 0)$; i.e. η is defined as the variance of the functional random variables.

Now, suppose that the Brownian paths $\{x_1, ..., x_d\} = \{B_{t_1}, ..., B_{t_d}\}$. The path x_{π_k} is generated in the order of $\boldsymbol{\pi}$ according to the Algorithm 4.1.1. Then, the algorithm of SC is defined as follows:

Algorithm 4.2.3. *Goal: construct decomposition matrix $\boldsymbol{A}$ according to the importance index array $\boldsymbol{\pi} = (\pi_1, ..., \pi_d)$, with the Brownian paths B_{t_π} generated in the order of $\boldsymbol{\pi}$.*
Step 1: For $i = 1 : d$
Construct $B_{t_{\pi_i}}$ according to 4.1.1.
End For
Step 2: Construct the matrix $[B_{t_1}, B_{t_2}, ..., B_{t_d}]^T = \boldsymbol{AZ}$, where $\boldsymbol{Z} = (z_1, ..., z_d)^T$.
Step 3: Return $\boldsymbol{A}$

Note that the SC method coincides with the BB construction when the importance index array $\boldsymbol{\pi}$ is given by $(d, d/2, d/4, 3d/4, \dots)$ assuming d is a power of 2.

4.2.4 Mixture Control (MIXC)

The purpose of the method of mixture control (MIXC) is to integrate both PC and SC methods. Suppose $\boldsymbol{\pi} = (\pi_1, ..., \pi_d)$ corresponds to the importance index array as defined in SC and let k_{MIXC} be the number of directions wish to be controlled by SC. Ideally k_{MIXC} should be small relative to d. Under the proposed MIXC, the first k_{MIXC} columns of $\boldsymbol{A}$ are determined using the method of SC (i.e. using Algorithm 4.2.3). The remaining columns of $\boldsymbol{A}$ are then optimally determined by applying singular value decomposition (i.e. using Algorithm 4.2.1 with $s = k_{MIXC} + 1$). Note that in the special case that $f_1 = x_1$, ..., $f_d = x_d$, MIXC becomes the mixture of PCA and BB.

Definition 4.2.1. *(Full Variability)*
Suppose $\boldsymbol{x} = (x_1, ..., x_d)^T = \boldsymbol{A}\boldsymbol{z} \sim N_d(\boldsymbol{0}, \boldsymbol{\Sigma})$*, where* $\boldsymbol{A}$ *is an arbitrary decomposition of* $\boldsymbol{\Sigma}$*, and* $\boldsymbol{z} \sim N_d(\boldsymbol{0}, \boldsymbol{I})$*. We define that the random variable or the Brownian path* x_i *keeps full variability under* $\boldsymbol{A}$ *up to dimension* k *for* $k \in [1, d]$ *if the elements on the* i^{th} *row are all zeros after the* k^{th} *column; i.e.*

$$A_{i,k} \neq 0, A_{i,k+1} = A_{i,k+2} ... = A_{i,d} = 0,$$

where $A_{i,j}$ *is defined as the element in the* i^{th} *row and* j^{th} *column of the matrix* $\boldsymbol{A}$*.*

Then, we have the following proposition
Proposition 4.2.1. *Suppose* $\boldsymbol{x} = (x_1, ..., x_d)^T = \boldsymbol{A}\boldsymbol{z} \sim N_d(\boldsymbol{0}, \boldsymbol{\Sigma})$*, where* $\boldsymbol{A}$ *is a decomposition of* $\boldsymbol{\Sigma}$ *generated by SC method, and* $\boldsymbol{z} \sim N_d(\boldsymbol{0}, \boldsymbol{I})$*. Furthermore, let* $\boldsymbol{x_\pi} = (x_{\pi_1}, ..., x_{\pi_d})$ *such that* $\{\pi_1, ..., \pi_d\}$ *are the order of path generation under SC. Then, we have the following property:* x_{π_i} *keeps full variability under* A *up to dimension* i*.*

Proof: It follows from Algorithms 4.2.3 and 4.1.1 that the generation of the Brownian path x_{π_i} only depends on the random variables $z_1, ..., z_i$. Since

$$x_{\pi_i} = \boldsymbol{A}_{\pi_i.}\boldsymbol{z} \tag{4.23}$$

where $\boldsymbol{A}_{k\cdot}$ denotes the k^{th} row of the matrix $\boldsymbol{A}$, and the elements on the π_i^{th} row of $\boldsymbol{A}$ are constructed to be all zeros after the i^{th} column. □

As we have seen in Section 4.2.2, the method of PC is to control the column directions such that $\|f\|$ is maximized for just a few dimensions. i.e. The first few columns of A_f dominate the function. As we have seen in Section 4.2.3, the method of SC is to control the order of generating Brownian paths according to the importance function of functional random variables. Many path dependent options, such as lookback option, barrier option, and Asian option, rely heavily on the underlying asset paths, so keep the full variability of important variables up to effective dimension is important to reduce the error bound of QMC since the discrepancy of the first few dimensions is much lower than the discrepancy of all d-dimensional points. In order to get both benefit of SC and PC, we propose the MIXC method so that some small number of variables will be kept in full variability and the rest of dimensions will be treated as equally. The algorithm is defined as follows:

Algorithm 4.2.4. *Goal: suppose k_{MIXC} is the number directions controlled by SC with $\boldsymbol{\pi} = (\pi_1, ..., \pi_d)$ being the importance index array. This algorithm maintains full variability up to k_{MIXC} important variables and maximizes the rest of functions by the functional variability.*
Step 1: Construct $\boldsymbol{A}$ according to Algorithm 4.2.3.
Step 2: Define $\hat{\boldsymbol{A}}_{\boldsymbol{k_{MIXC}}}$ as the first k columns of $\boldsymbol{A}$. (i.e. keep the first k_{MIXC} columns)
Step 3: Get $\hat{\boldsymbol{A}}_{f_{remaining}}$ by the SVD of $\boldsymbol{J}(\boldsymbol{\Sigma} - \hat{\boldsymbol{A}}_{k_{MIXC}}\hat{\boldsymbol{A}}_{k_{MIXC}}^T)\boldsymbol{J}^T$. (i.e. by Algorithm 4.2.1 with $s = k_{MIXC} + 1$)
Step 4: $\hat{\boldsymbol{A}}_{remaining} = \boldsymbol{J}^{-1}\hat{\boldsymbol{A}}_{f_{remaining}}$
Step 5: Return $\hat{\boldsymbol{A}} = \left[\hat{\boldsymbol{A}}_{k_{MIXC}}; \hat{\boldsymbol{A}}_{remaining}\right]$

4.2.5 Goodness of Functional Decomposition Criterion (Special Case)

Now we define some criterion to compare the goodness of functional decomposition. Suppose $\boldsymbol{x} = (x_1, ..., x_d)^T \in N_d(\mathbf{0}, \boldsymbol{\Sigma})$, and we consider the special case in which the function f can be expressed in term of the sum of the correlated random variables $\{f_1(x_1), .., f_d(x_d)\}$, i.e.

$$f(\boldsymbol{x}) = f_1(x_1) + f_2(x_2) + ... + f_d(x_d). \tag{4.24}$$

Remark 4.2.3. *In the case that the price at time t_i only depends on a single random variable x_i and the payoff function f depends on the summation of prices at each time t_i for $i = 1, ..., d$, we would consider the form* (4.24)*. For example, let us consider pricing an Asian option under classical Black-Scholes model, we would consider $f_i(x_i) = S_0 \exp((r - \frac{\sigma^2}{2})t_i + \sigma x_i)$. Some more usefulness of decomposing the structure in the form* (4.24) *will be discussed in Section 6.2.*

Furthermore, we let $\boldsymbol{Y} = (f_1(x_1), ..., f_d(x_d))^T$, and obviously, $\{f_1(x_1)\ ,\ ...,\ f_d(x_d)\}$ are the correlated functional random variables containing the random variable x_i's.

By applying the first-order vector Taylor expansion to each of the function f_i, for $i = 1, ..., d$, around an arbitrary vector $\boldsymbol{x} = \hat{\boldsymbol{x}} + \Delta\boldsymbol{x}$:

$$\boldsymbol{Y} \approx \boldsymbol{Y}(\hat{\boldsymbol{x}}) + \nabla\boldsymbol{Y}(\hat{\boldsymbol{x}})(\boldsymbol{x} - \hat{\boldsymbol{x}}). \tag{4.25}$$

Then, applying the Delta method, the covariance matrix of Y is defined as follows:

$$\boldsymbol{\Sigma}_f = Var(\boldsymbol{Y}) = \boldsymbol{J}\boldsymbol{\Sigma}\boldsymbol{J}^T$$

where under our assumed special case, the Jacobian matrix $\boldsymbol{J}$ is a diagonal matrix; i.e.

$$\boldsymbol{J} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & 0 & \cdots & 0 \\ 0 & \frac{\partial f_2}{\partial x_2} & \cdots & 0 \\ \vdots & \ddots & \vdots & \\ 0 & \cdots & 0 & \frac{\partial f_d}{\partial x_d} \end{pmatrix}.$$

Thus, our $Var(f)$ is equal to the summation of all elements in $Var(\boldsymbol{Y})$; i.e. $Var(f) = \sum_{i=1}^{d}\sum_{j=1}^{d}(\boldsymbol{J\Sigma J}^T)_{i,j}$.

We now propose three criteria to evaluate our three directional control methods. All these methods attempt to find the optimal decomposition matrix $\boldsymbol{A}$ such that $\boldsymbol{J\hat{A}_k\hat{A}_k}^T\boldsymbol{J}^T$ is maximized for low dimensions, where $\boldsymbol{\hat{A}_k}$ denotes the first k columns of $\boldsymbol{A}$ for $k = 1, ..., d,$. We propose the following three criteria to evaluate the goodness of functional decomposition according to our three directional methods which control the column directions, row directions, and mixed directions of the decomposition matrix $\boldsymbol{A}$, respectively. In what follows, we define

$$\begin{aligned} \boldsymbol{x} &= \boldsymbol{Az},\ \boldsymbol{z} \sim N_d(\boldsymbol{0}, \boldsymbol{I}), && (4.26)\\ \textit{and } \hat{\boldsymbol{x}} &= \boldsymbol{\hat{A}_k\hat{z}_k},\ \hat{\boldsymbol{z}}_{\boldsymbol{k}} \sim N_k(\boldsymbol{0}, \boldsymbol{I}). && (4.27) \end{aligned}$$

where $\boldsymbol{A}$ is an arbitrary decomposition matrix of $\boldsymbol{\Sigma}$, $\boldsymbol{\hat{A}_k}$ denoting the first k columns of $\boldsymbol{A}$ is a d by k matrix, and $\hat{\boldsymbol{z}}_{\boldsymbol{k}}$ denotes the vector of first k random variables of $\boldsymbol{z}$. The parameter $\alpha \in [0, 1]$ and is typically set to a value close to one.

1. Criterion: Projection Control (PC)
   The PC method controls the column directions of decomposition matrix by seeking minimum k such that the following inequality is satisfied:

$$\begin{aligned} \min\{k : Var(f(\hat{\boldsymbol{x}})) &= Var(f_1(\hat{x}_1) + f_2(\hat{x}_2) + \cdots + f_d(\hat{x}_d)) \\ &= Var(f_1(\boldsymbol{\hat{A}_{1.,k}\hat{z}_k}) + f_2(\boldsymbol{\hat{A}_{2.,k}\hat{z}_k}) + ... + f_d(\boldsymbol{\hat{A}_{d.,k}\hat{z}_k})) \\ &> \alpha Var(f)\}, \end{aligned}$$

   where $\boldsymbol{\hat{A}_{i.,k}}$ denotes the first k elements in the i^{th} row of the decomposition matrix $\boldsymbol{A}$, and $\hat{\boldsymbol{z}}_{\boldsymbol{k}} = (z_1, ..., z_k)^T \sim N_k(\boldsymbol{0}, \boldsymbol{I})$. The optimal algorithm of PC method according to the functional variability is proposed in Algorithm 4.2.1.

2. Criterion: Sequential Control (SC)
   SC method controls the row generation sequences of decomposition matrix by seeking

the minimum k such that the following inequality is satisfied:

$$\begin{aligned}\min\{k : Var(f(\hat{\boldsymbol{x}})) &= Var(f_1(\hat{x}_1) + f_2(\hat{x}_2) + \cdots + f_d(\hat{x}_d)) \\ &= Var(f_1^*(\hat{\boldsymbol{A}}^*_{\mathbf{1.,1}}\hat{\boldsymbol{z}}_{\mathbf{1}}) + f_2^*(\hat{\boldsymbol{A}}^*_{\mathbf{2.,2}}\hat{\boldsymbol{z}}_{\mathbf{2}}) + ... + f_k^*(\hat{\boldsymbol{A}}^*_{\boldsymbol{k.,k}}\hat{\boldsymbol{z}}_{\boldsymbol{k}}) \\ &+ f_{k+1}^*(\hat{\boldsymbol{A}}^*_{\boldsymbol{k+1.,k}}\hat{\boldsymbol{z}}_{\boldsymbol{k}}) + .. + f_d^*(\hat{\boldsymbol{A}}^*_{\boldsymbol{d.,k}}\hat{\boldsymbol{z}}_{\boldsymbol{k}})) \\ &> \alpha Var(f)\}.\end{aligned}$$

Here $f_1^*, f_2^*, ..., f_d^*$ are the ordered functions according to the importance of each additive decompositions term of the function f, $\boldsymbol{A}^*_{\boldsymbol{i.}}$ is the ordered i^{th} row of the decomposition matrix $\boldsymbol{A}$ according to the order of f_i's for $i = 1, 2, ..., d$, and $\hat{\boldsymbol{z}}_{\boldsymbol{i}}$ denotes the vector of first i random variables of $\boldsymbol{z}$, for $i = 1, ..., k$. i.e. $\hat{\boldsymbol{z}}_{\boldsymbol{i}} \sim N_i(\mathbf{0}, \boldsymbol{I})$, for $i = 1, ..., k$. This method keeps the full variability up to dimension k for the first indexed/ordered k functions. The algorithm of SC method is proposed in Algorithm 4.2.3.

3. Criterion: Mixture Control (MIXC)
   MIXC method controls the row generation sequences of decomposition matrix first and the column directions of decomposition matrix by seeking minimum k such that the following inequality is satisfied:

$$\begin{aligned}\min\{k : Var(f(\hat{\boldsymbol{x}})) &= Var(f_1(\hat{x}_1) + f_2(\hat{x}_2) + \cdots + f_d(\hat{x}_d)) \\ &= Var(f_1^*(\hat{\boldsymbol{A}}^*_{\mathbf{1.,1}}\hat{\boldsymbol{z}}_{\mathbf{1}}) + f_2^*(\hat{\boldsymbol{A}}^*_{\mathbf{2.,2}}\hat{\boldsymbol{z}}_{\mathbf{2}}) \\ &+ ... + f_{k_{MIXC}}^*(\hat{\boldsymbol{A}}^*_{\boldsymbol{k_{MIXC}.,k_{MIXC}}}\hat{\boldsymbol{z}}_{\boldsymbol{k_{MIXC}}}) \\ &+ f_{(k_{MIXC}+1)}^*(\hat{\boldsymbol{A}}^*_{\boldsymbol{k_{MIXC}+1.,k}}\hat{\boldsymbol{z}}_{\boldsymbol{k}}) + .. + f_{(d)}^*(\hat{\boldsymbol{A}}^*_{\boldsymbol{d.,k}}\hat{\boldsymbol{z}}_{\boldsymbol{k}})) \\ &> \alpha Var(f)\},\end{aligned}$$

   where k_{MIXC} is the sequential controlling dimension (i.e. the number of rows we need to control to keep the full variability), and k is the effective dimension (according to our proposed delta dimension) we need to control. So, $k - k_{MIXC}$ is the projection controlling dimension (i.e. the number of columns we need to control to minimized the residual functional variability). $f_1^*, f_2^*, ..., f_d^*$ are the ordered functions according to

the importance of each additive decompositions term of the function f (e.g. for Asian option, index is the variability of $f_i's$; for lookback option, the index is the expectation of $f_i's$), $\hat{\boldsymbol{A}}^*_{\boldsymbol{i},\boldsymbol{j}}$ are the first j elements of the ordered i^{th} row of the decomposition matrix $\boldsymbol{A}$ according to the order of the importance index function of f_i's for $i = 1, 2, ..., d$, $j = 1, 2, .., k$, , and $\hat{\boldsymbol{z}}_{\boldsymbol{i}}$ denotes the vector of first i random variables of $\boldsymbol{z}$, for $i = 1, 2, ..., k_{MIXC}, k$. i.e. $\hat{\boldsymbol{z}}_{\boldsymbol{i}} \sim N_i(\boldsymbol{0}, \boldsymbol{I})$. This method keeps the full variability for the first indexed/ordered k_{MIXC} functions by SC, and maximize the functional variability for the rest of dimensions by PC. The algorithm of MIXC method is proposed in Algorithm 4.2.4. Note that when $k_{MIXC} = 0$, MIXC reduces to PC, and when $k_{MIXC} = d$, MIXC reduces to SC.

4.2.6 Goodness of Functional Decomposition Criterion (General)

Suppose $\boldsymbol{Y} = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ..., f_d(\boldsymbol{x}))^T$, where $\boldsymbol{x}$ has mean 0 and covariance matrix $\boldsymbol{\Sigma}$, and suppose further there exists a function ψ such that

$$f = \psi(f_1(\boldsymbol{x}), ..., f_d(\boldsymbol{x})).$$

(i.e. The variability of the function f depends on the covariance matrix of $\boldsymbol{Y}$.) We know that $\boldsymbol{Y}$ can be also written as $\boldsymbol{Y} = (f_1(\boldsymbol{Az}), f_2(\boldsymbol{Az}), ..., f_d(\boldsymbol{Az}))^T$, where $\boldsymbol{A}$ is an arbitrary decomposition of $\boldsymbol{\Sigma}$ and $\boldsymbol{z} \sim N_d(\boldsymbol{0}, \boldsymbol{\Sigma})$. Our purpose is to find the optimal decomposition of $\boldsymbol{\Sigma}$ according to the covariance matrix $Var(\boldsymbol{Y})$ for the first k dimensions. We define

$$\hat{\boldsymbol{Y}} = [f_1(\hat{\boldsymbol{A}}_{\boldsymbol{k}}\hat{\boldsymbol{z}}_{\boldsymbol{k}}), f_2(\hat{\boldsymbol{A}}_{\boldsymbol{k}}\hat{\boldsymbol{z}}_{\boldsymbol{k}}), ..., f_d(\hat{\boldsymbol{A}}_{\boldsymbol{k}}\hat{\boldsymbol{z}}_{\boldsymbol{k}})]^T,$$

where $\hat{\boldsymbol{z}}_{\boldsymbol{k}} \sim N_k(\boldsymbol{0}, \boldsymbol{I})$ and $\hat{\boldsymbol{A}}_{\boldsymbol{k}}$ denotes the first k columns of $\boldsymbol{A}$.

By the Delta method, we have

$$\boldsymbol{\Sigma}_f = Var(\boldsymbol{Y}) = \boldsymbol{J\Sigma J}^T,$$

where $\boldsymbol{J}$ is the Jacobian matrix of f w.r.t $\boldsymbol{x}$, i.e.

$$\begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_d} \\ \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_2}{\partial x_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_d}{\partial x_1} & \cdots & \frac{\partial f_d}{\partial x_d} \end{pmatrix},$$

and

$$\hat{\boldsymbol{\Sigma}}_f^{(k)} = Var(\hat{\boldsymbol{Y}}) = \boldsymbol{J}\hat{\boldsymbol{\Sigma}}^{(k)}\boldsymbol{J}^T,$$

where $\hat{\boldsymbol{\Sigma}}^{(k)} = \hat{\boldsymbol{A}}_{\boldsymbol{k}}\hat{\boldsymbol{A}}_{\boldsymbol{k}}{}^T$ denotes the covariance matrix up to dimension k, and $\hat{\boldsymbol{A}}_{\boldsymbol{k}}$ denotes the first k columns of $\boldsymbol{A}$. (The special case defined in Section 4.2.5 is $\boldsymbol{Y} = (f_1(x_1), f_2(x_2), ..., f_d(x_d))^T$ where the Jacobian matrix is diagonal matrix.)

Remark 4.2.4. *$\boldsymbol{J}$ is not necessarily a symmetric matrix, but the decomposed functional covariance matrix $\boldsymbol{\Sigma}_f$ is always a symmetric matrix. However, rank($\boldsymbol{\Sigma}_f$) is not necessary equal to d if rank($\boldsymbol{J}$) $\neq d$; e.g. the European option only depends on x_d. In this case, the total dimension we need to control will be equivalent to rank($\boldsymbol{\Sigma}_f$). So, f_i's are the random variables which try to keep the most information of the price at time t_i's, and in the special case that f_i's can not contain more information than x_i's, we simple assume that $f_i = x_i$, for $i = 1, ..., d$. The purpose of our DC method is to do dimension reduction depending on the random variables f_i's instead of x_i's, for $i = 1, ..., d$.*

Then, following the definitions of (4.26) and (4.27), our DC methods control the directions of decomposition matrix by seeking minimum k such that the following inequality is satisfied:

$$\begin{aligned} \min\{k : Var(f(\hat{\boldsymbol{x}})) &= Var(\psi(\hat{\boldsymbol{Y}})) \\ &= Var(\psi(f_1(\hat{\boldsymbol{A}}_{\boldsymbol{k}}\hat{\boldsymbol{z}}_{\boldsymbol{k}}), f_2(\hat{\boldsymbol{A}}_{\boldsymbol{k}}\hat{\boldsymbol{z}}_{\boldsymbol{k}}), ..., f_d(\hat{\boldsymbol{A}}_{\boldsymbol{k}}\hat{\boldsymbol{z}}_{\boldsymbol{k}}))) \\ &> \alpha Var(f)\}, \end{aligned}$$

where $\hat{\boldsymbol{A}}_{\boldsymbol{k}}$ denoting the first k columns of $\boldsymbol{A}$ is a d by k matrix, and $\hat{\boldsymbol{z}}_{\boldsymbol{k}}$ denotes the vector of first k random variables of $\boldsymbol{z}$. i.e. $\hat{\boldsymbol{z}}_{\boldsymbol{k}} \sim N_k(\boldsymbol{0}, \boldsymbol{I})$.

4.2.7 Variability Comparison Criteria

It is customary to use the explained variability to assess the quality of the decomposition matrix of $\boldsymbol{\Sigma}$. However, the decomposition according to our DC method depends on the structure of functional covariance matrix $\boldsymbol{\Sigma}_f$, so we introduce some modified criteria to compare the variability of the decomposition matrix. As we have defined earlier that $\boldsymbol{\Sigma} = \boldsymbol{A}\boldsymbol{A}^T$, $\hat{\boldsymbol{\Sigma}}^{(k)} = \hat{\boldsymbol{A}}_{\boldsymbol{k}}\hat{\boldsymbol{A}}_{\boldsymbol{k}}{}^T$, $\boldsymbol{\Sigma}_f = \boldsymbol{J}\boldsymbol{\Sigma}\boldsymbol{J}^T$, and $\hat{\boldsymbol{\Sigma}}_f^{(k)} = \boldsymbol{J}\hat{\boldsymbol{A}}_{\boldsymbol{k}}\hat{\boldsymbol{A}}_{\boldsymbol{k}}{}^T\boldsymbol{J}^T$, where $\boldsymbol{J}$ is the Jacobian matrix of f. Then, we define the following variability comparison criteria:

1. Explained variability:

   Explained variability up to k-th column of $\boldsymbol{A}$ is given by

   $$\begin{aligned} \hat{E}_k &= \frac{\|\boldsymbol{A}_{\boldsymbol{.1}}\|^2 + \|\boldsymbol{A}_{\boldsymbol{.2}}\|^2 + \cdots + \|\boldsymbol{A}_{\boldsymbol{.k}}\|^2}{\|\boldsymbol{A}_{\boldsymbol{.1}}\|^2 + \|\boldsymbol{A}_{\boldsymbol{.2}}\|^2 + \cdots + \|\boldsymbol{A}_{\boldsymbol{.d}}\|^2} \\ &= \frac{\|\boldsymbol{A}_{\boldsymbol{.1}}\|^2 + \|\boldsymbol{A}_{\boldsymbol{.2}}\|^2 + \cdots + \|\boldsymbol{A}_{\boldsymbol{.k}}\|^2}{trace(\boldsymbol{\Sigma})} \qquad (4.28) \\ &= \frac{\sum_{i=1}^{k} \boldsymbol{\Sigma}_{i,i}}{trace(\boldsymbol{\Sigma})}, \qquad (4.29) \end{aligned}$$

   where $\boldsymbol{A}_{\boldsymbol{.i}}$ denotes the i^{th} column of $\boldsymbol{A}$, and $\boldsymbol{\Sigma}_{i,i}$ denotes the i^{th} diagonal element in $\boldsymbol{\Sigma}$ for $i = 1, ..., d$. This is a commonly used criterion for comparing the effectiveness of the explained variability for a given decomposition matrix of a covariance matrix.

2. Residual Variability r_k: Suppose $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{z}$ and $\boldsymbol{x}_k = \hat{\boldsymbol{A}}_{\boldsymbol{k}}\hat{\boldsymbol{z}}_{\boldsymbol{k}}$, where $\boldsymbol{z} \sim N_d(\boldsymbol{0}, \boldsymbol{I})$, $\hat{\boldsymbol{z}}_{\boldsymbol{k}} \sim N_k(\boldsymbol{0}, \boldsymbol{I})$, $\hat{\boldsymbol{A}}_{\boldsymbol{k}}$ denotes the first k columns of $\boldsymbol{A}$, and $\hat{\boldsymbol{z}}_{\boldsymbol{k}}$ denotes the vector of first k random variables of $\boldsymbol{z}$. Then, we define the residual σ_x^2 up to dimension k as

   $$\begin{aligned} r_k &= Var(x_1 + x_2 + \cdots + x_d) - Var(\hat{x}_1 + \hat{x}_2 + \cdots + \hat{x}_d) \\ &= \sum_{i=1}^{d}\sum_{j=1}^{d} \boldsymbol{\Sigma}_{ij} - \sum_{i=1}^{d}\sum_{j=1}^{d} \hat{\boldsymbol{\Sigma}}_{ij}^{(k)}. \end{aligned}$$

   Remark 4.2.5. *According to Lemma 3.4.1, 3.4.2 and 3.4.3, we have*

   $$\sum_{i=1}^{d}\sum_{j=1}^{d} \boldsymbol{\Sigma}_{ij} \geq \sum_{i=1}^{d}\sum_{j=1}^{d} \hat{\boldsymbol{\Sigma}}_{ij}^{(k)} \geq 0. \qquad (4.30)$$

This criterion gives the same information as criterion 1 if we don't consider the covariance between x_i and x_j, for $\forall i \neq j$. i.e.

$$\begin{aligned} & [Var(x_1) + Var(x_2) + \cdots + Var(x_d)] - [Var(\hat{x}_1) + Var(\hat{x}_2) + \cdots + Var(\hat{x}_d))] \\ = & \sum_{i=1}^{d} \mathbf{\Sigma}_{i,i} - \sum_{i=1}^{d} \hat{\mathbf{\Sigma}}_{i,i}^{(k)} \\ = & \ trace(\mathbf{\Sigma}) - trace(\mathbf{\Sigma})\hat{E}_k \\ = & \ trace(\mathbf{\Sigma})(1 - \hat{E}_k). \end{aligned}$$

We introduce this criterion so that we can highlight the importance of our own criterion 3.

3. Residual Functional Variability R_k:
   Suppose $\boldsymbol{Y} = (f_1(\boldsymbol{x}), ..., f_d(\boldsymbol{x}))^T$ and $\hat{\boldsymbol{Y}} = (f_1(\hat{\boldsymbol{x}}), ..., f_d(\hat{\boldsymbol{x}})^T$, where $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ are same as what we defined in criterion 1. This criterion is similar to the above criterion except that $\mathbf{\Sigma}_f$ is used, instead of $\mathbf{\Sigma}$:

$$\begin{aligned} R_k &= Var(Y_1 + Y_2 + \cdots + Y_d) - Var(\hat{Y}_1 + \hat{Y}_2 + \cdots + \hat{Y}_d) \\ &= \sum_{i=1}^{d}\sum_{j=1}^{d} (\mathbf{\Sigma}_f)_{ij} - \sum_{i=1}^{d}\sum_{j=1}^{d} \left(\hat{\mathbf{\Sigma}}_f^{(k)}\right)_{ij}. \end{aligned}$$

   Remark 4.2.6. *According to Lemma 3.4.1, 3.4.2 and 3.4.3, we have*

$$\sum_{i=1}^{d}\sum_{j=1}^{d} (\mathbf{\Sigma}_f)_{ij} \geq \sum_{i=1}^{d}\sum_{j=1}^{d} \left(\hat{\mathbf{\Sigma}}_f^{(k)}\right)_{ij} \geq 0. \tag{4.31}$$

   The smaller R_k is, the better the decomposition method is for the first k components in the case of maximizing the variability of the random variables $(f_1, ..., f_d)$.

Remark 4.2.7. *In the special case that $f_1 = x_1$, ..., $f_d = x_d$, our new criteria for σ_f^2 becomes the same as the criteria of residual σ_x^2.*

4.2.8 Relation of DC with Other Methods

In the special case that if we let $f_1(x_1) = x_1, ..., f_d(x_d) = x_d$, and $\psi = f$ then

$$f(x_1, ., , , x_d) = \psi(f_1(x_1), ..., f_d(x_d)) = \psi(x_1, ..., x_d), \tag{4.32}$$

where controlling the decomposition of covariance matrix $\mathbf{\Sigma}_f$ is the same as the decomposition of covariance matrix $\mathbf{\Sigma}$. In this case, our projection control method will have the same effect as the orthogonal transformation and the optimal solution in the sense of the explained variability is the same as PCA method, our sequential control method will have the same effect as the permutation of Brownian motion, and the optimal solution in the sense of the explained variability is the new Brownian bridge (NBB) method proposed by Lin and Wang (2008). On the other hand, the general optimization of projection control is the same as LT method in the special case that $f = f_1(x_1) + ... + f_d(x_d)$. i.e $\boldsymbol{Y} = (f_1(x_1), ..., f_d(x_d))^T$, where the Jacobian matrix is defined as the derivatives of $\boldsymbol{Y}$ with respect to $\boldsymbol{x}$ is a diagonal matrix with (i, i)-th entry given by $\frac{\partial f_i}{\partial x_i}$; i.e. $diag(\boldsymbol{J}) = \nabla f(\boldsymbol{x})$.

4.3 Time Complexity

While the possible decompositions of $\mathbf{\Sigma}$ provide a framework for the algorithms to generate the Brownian paths, they are of limited practical value because the matrix-vector multiplication is comparatively slow for all but small values of d. This is the case since the general matrix-vector multiplication uses $O(d^2)$ floating point operations (flops). By constructions, we know that the random walk (or standard) construction and BB construction only use $O(d)$ flops to generate one Brownian path. Scheicher (2007) showed that PCA can be computed using fast sine transform, and thereby using $O(d \log(d))$ flops. Lobache (2012) showed that orthogonal transformation using fast Fourier transforms(FFT), which computes the discrete Fourier transform (DFT) and produces exactly the same result as evaluating the DFT definition directly, will take $O(d \log(d))$ flops.

For our projection control method, evaluating $\boldsymbol{J}^{-1}\boldsymbol{A_f}$, where $\boldsymbol{J}$ is the Jacobian matrix and $\boldsymbol{A_f}$ is the decomposition for functional covariance matrix $\boldsymbol{\Sigma}_f$, can be done in $O(d^3)$ flops. However, if $\boldsymbol{J}$ is a diagonal matrix, the time complexity could be improved to be $O(d^2 \log(d))$ flops which is the same as linear transformation. For our sequential control method, the time complexity is the same the BB construction, which take $O(d)$ flops. Using mixture control method which controls the fixed number of dimensions by SC method and using PC method to control the rest of dimensions will takes $O(d^2)$ flops at most. However, if we only control the first few dimensions and using singular value decomposition of $\boldsymbol{\Sigma}$ for the rest of dimensions will take $O(d \log(d))$, or using forward or Brownian bridge construction for the rest of dimensions will takes $O(\max(k^2, d))$ flops, where k is the number of directions we want to control. The partial controlling which only controls the effective dimensions is still under investigation. However, in the case of evaluating high dimensional integrals, the number of simulation paths N is less than d^2 if d is sufficiently large (e.g. $d \geq 360$), so in that case, applying SC and MIXC methods will be more important than our PC methods.

Considering that our partition of the payoff structure is suitable for parallel computing, for example, we could use the machine m_i to evaluate the decomposed function f_i for $i = 1, ..., d$, which will be discussed in the section of future research, we would eliminate the time complexity for functional evaluation by the simulation of multiple machines simultaneously. In this case, we use the slow construction upper bound $O(d^2)$ to improve the even more slower evaluation time of the functions with MC or QMC methods, which takes $O(dmN)$, where d is the dimensionality, N is the number of input sequences, and m is the number of replication. And considering that N is always much larger than d^2, the trade-off is extremely worthwhile.

Chapter 5

Numerical Examples

The main purpose of this chapter is to examine the efficiency of our proposed directional control (DC) methods relative to other QMC-based methods. The relative efficiency of these methods are accessed by examining the explained variability, residual functional variability and the variance reduction ratio using a wide range of high-dimensional European derivative securities.

Recall that under the Black-Scholes (BS) model, the risk-neutral process of the underlying asset is given by

$$dS_t = rS_t\,dt + \sigma S_t\,dB_t, \tag{5.1}$$

where r is the risk-free interest rate, σ is the volatility and B_t is the standard Brownian motion. Suppose $h(\boldsymbol{S}) = h(S_1, \ldots, S_d)$ denotes the payoff function of a derivative security at maturity T years, which depends on the asset prices $S_j := S_{t_j}$ at equally spaced times $t_j = j\Delta t$ for $j = 1, \ldots, d$ and $\Delta t = T/d$. According to the option pricing theory, the value of the financial derivative at $t = 0$ is $\mathbb{E}\left[e^{-rT}h(\mathbf{S})\right]$, where $\mathbb{E}\left[\cdot\right]$ is the expectation under the risk-neutral measure.

Let $\boldsymbol{x} := (x_1, \ldots, x_d)^T \sim N_d(\mathbf{0}, \boldsymbol{\Sigma})$; i.e, $\boldsymbol{x}$ is normally distributed with mean $\mathbf{0}$ and covari-

ance matrix $\mathbf{\Sigma}$ with its entry given by

$$\Sigma_{ij} = \min(t_i, t_j) = \Delta t \, \min(i, j), \tag{5.2}$$

the payoff function $h(\boldsymbol{S})$ can be expressed in term of $\boldsymbol{x}$ as

$$h(\boldsymbol{S}) = h(\exp(\mu_1 + \sigma x_1), \ldots, \exp(\mu_d + \sigma x_d)) := g(\boldsymbol{x}), \tag{5.3}$$

where $\mu_j = \log S_0 + (r - \sigma^2/2)\, t_j$. Note that we have redefined $h(\mathbf{S})$ as $g(\boldsymbol{x})$ to emphasize the explicit role of $\boldsymbol{x}$.

For a given PGM $\boldsymbol{A}$ satisfying $\boldsymbol{A}\boldsymbol{A}^T = \mathbf{\Sigma}$ and a uniform vector $\boldsymbol{u} = (u_1, \ldots, u_d) \in [0, 1)^d$, each x_i in the payoff function $g(\boldsymbol{x})$ can be simulated as

$$x_i = \boldsymbol{A}_{i\cdot}\boldsymbol{z}$$

where $\boldsymbol{z} = (z_1, \ldots, z_d)^T = (\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_d))^T$ and $\Phi^{-1}(\cdot)$ is the inverse of the standard normal distribution. Hence $\boldsymbol{z} \sim N_d(\mathbf{0}, I)$ where I is the $d \times d$ identity matrix and $\boldsymbol{u} = (u_1, \ldots, u_d)$ is the d-dimensional points in $[0, 1)^d$. The above equation suggests that for a given vector $\boldsymbol{z}$, the simulated stock price at time t_j differs depending on the choice of $\boldsymbol{A}$. This in turn may have important implication on the estimated prices of the derivative securities (which depend on the simulated asset prices).

In comparing various QMC-based methods, we use the following three criteria:

- explained variability,
- residual functional variability
- variance reduction ratio.

The first two criteria focus on the effectiveness of the decomposition matrices $\boldsymbol{A}$ by comparing their explained variability and the residual functional variability. Ultimately we are interested in the performance of using these decomposition matrices for simulating the option prices. In other words, we are interested in the QMC-based PGM involving these matrices. A wide range of options are used for this purpose. For each option example, each

of the decomposition matrix $\boldsymbol{A}$ is used to simulate the stock prices, estimate the option prices as well as estimate the variance of the option estimator. For the QMC-based methods, we use the scrambled Sobol sequences that are based on a random linear scramble combined with a random digital shift. The variance of the option estimator is estimated by M independent set of scrambled sequences of N points. On the other hand, an estimate of the variance of the option estimator for the Monte Carlo method is based on a $M \times N$ points. The ratio of the variance of the MC estimate to the variance of the QMC estimate therefore provides a measure of the efficiency of QMC relative to the MC method. A variance reduction ratio greater than one implies that the underlying QMC method is more efficient than the MC method.

The remaining sections are devoted to comparing the relative efficiency of the PGMs on a number of high-dimensional options.

5.1 Asian Options

The first example we will study is the discrete arithmetic Asian call option with its payoff function given by

$$h(\boldsymbol{S}) = \max\left(\frac{1}{d}\sum_{i=1}^{d} S_i, K\right).$$

By setting $f(\boldsymbol{x}) = f_1(x_1) + \cdots f_d(x_d)$, where $f_i = \frac{1}{d}\exp(\mu_i + \sigma x_i)$, then the above payoff function can be written in term of $g(\boldsymbol{x})$ as

$$g(\boldsymbol{x}) = \max(f(\boldsymbol{x}) - K, 0) = \max(f_1(x_1) + ... + f_d(x_d) - K, 0)$$

We assume $S_0 = 100, T = 1, r = 0.1, \sigma = 0.2$ for our base case parameter values. The efficiency of our proposed direction method is compared to five other methods, namely

1. Standard approach (STD) with $\boldsymbol{A}$ corresponds to the Cholesky decomposition of $\boldsymbol{\Sigma}$; see Section 4.1.1.

2. Principal component analysis (PCA) method of Acworth et al. (1998); see Section 4.1.2.

3. Brownian bridge (BB) construction of Moskowitz and Caflisch (1996); see Section 4.1.3.

4. New Brownian bridge (NBB) construction of Lin and Wang (2008) with the optimal permutation of Brownian bridge tabulated in Appendix A. See also Section 4.1.3.

5. Discrete cosine transform (DCT) of Leobacher (2012); see Section 4.1.5.

5.1.1 Explained Variability

Recall that in Section 4.2.7, the explained variability up to the first k dimensions is given by

$$\hat{E}_k = \frac{\|\hat{\boldsymbol{A}}_{\boldsymbol{k}}\|}{\|\boldsymbol{A}\|}.$$

By definition, we have $0 < \hat{E}_k \leq 1$. This quantity measures the effectiveness of $\boldsymbol{A}$ in dimension reduction. If the effective dimension is small, we should expect that $\hat{E}_k$ is approaching one for low values of k.

Recall also that our proposed directional method requires the specification of "direction". In the Asian option, a logical selection is to consider $f(\boldsymbol{x}) = f_1(x_1) + \cdots + f_d(x_d)$ so that using the delta method, we can determine $\Sigma_f = Var(f(\boldsymbol{x}))$. Therefore the first implementation is according to PC_0 and PC. The PC method is according to the algorithm 4.2.1 with covariance matrix of $\boldsymbol{\Sigma}_f = var(\boldsymbol{Y})$. In this case, PC method will be the same as LT method due to the diagonal matrix of Jacobian matrix of $\boldsymbol{Y}$ with respect to $\boldsymbol{x}$. On the other hand, the method label "PC_0" is also based on PC method except by assuming $f(\boldsymbol{x}) = x_1 + \cdots + x_d$. This implies that PC_0 collapse to the PCA construction.

Table 5.1.1 and Table 5.1.1 report the explained variability ratios for the first four and five dimensions for $d = 4$ and $d = 16$, respectively.

Component	STD	PCA	BB	NBB	DCT	PC_0	PC
1	40.00	82.91	75.00	76.67	80.15	82.91	82.71
2	70.00	92.91	90.00	86.67	90.78	92.91	92.77
3	90.00	97.17	95.00	95.00	96.00	97.17	97.05
4	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Table 5.1: Cumulative explained variability ratios (in percentage) for the first four dimensions ($d = 4$)

Component	STD	PCA	BB	NBB	DCT	PC_0	PC
1	11.76	81.19	68.75	75.12	80.37	81.19	79.46
2	22.79	90.27	84.56	84.07	89.41	90.27	88.97
3	33.09	93.58	88.60	89.71	92.75	93.58	92.58
4	42.65	95.29	89.71	92.03	94.51	95.29	94.48
5	51.47	96.36	90.07	94.36	95.63	96.36	95.65

Table 5.2: Cumulative explained variability ratios (in percentage) for the first 5 dimensions ($d = 16$)

Using the explained variability as a criterion, we draw the following observations:

- The ineffectiveness of the STD method in dimension reduction is clearly highlighted. The cumulative explained variability ratios are the lowest for STD method.
- PCA construction is the most effective, as indicated by the highest ratios. This is not surprising since the decomposition matrix $\boldsymbol{A}$ under the PCA is designed to optimally maximize the explained variability for each successive dimension.
- Both PCA and PC_0 have the same ratios. This again is not surprising since PC_0 is designed to reproduce PCA. This confirms that our proposed direction control method is able to recover PCA, depending on the choice of the "direction".

- In term of the explained variability, other methods such as BB, NBB, DCT and PC are inferior to PCA, although they are rather close to the optimal values of PCA.

5.1.2 Residual Functional variability

Recall that in Section 4.2.7, the residual functional variability is defined as

$$R_k = \sum_{i=1}^{d}\sum_{j=1}^{d} (\mathbf{\Sigma}_f)_{ij} - \sum_{i=1}^{d}\sum_{j=1}^{d} \left(\hat{\mathbf{\Sigma}}_f^{(k)}\right)_{ij}.$$

Tables 5.1.2, 5.1.2, and 5.1.2 depict the residual functional variabilities for $d = 16, d = 64$ and $d = 128$.

Component	STD	PCA	BB	NBB	DCT	PC
1	86.91	0.08	15.61	8.27	1.43	2.22
2	74.64	0.08	4.31	4.91	1.36	0.31
3	63.24	0.01	3.60	1.54	1.08	0.11
4	52.74	0.01	3.54	1.28	0.98	0.05
5	43.17	0.00	3.53	0.62	0.87	0.03

Table 5.3: Residual functional variability for first 5 dimensions ($d = 16$)

Component	STD	PCA	BB	NBB	DCT	PC
1	97.85	4.25	7.71	9.16	4.18	1.50
2	95.70	0.53	4.15	8.89	0.74	0.29
3	93.55	0.10	4.13	0.94	0.26	0.13
4	91.40	0.03	4.13	0.93	0.21	0.07
5	89.26	0.01	4.13	0.79	0.18	0.04

Table 5.4: Residual functional variability for first 5 dimensions ($d = 64$)

Component	STD	PCA	BB	NBB	DCT	PC
1	99.09	9.38	4.23	14.21	9.29	0.81
2	98.18	2.18	3.22	14.20	2.27	0.16
3	97.27	0.66	3.22	1.07	0.71	0.07
4	96.36	0.24	3.22	1.07	0.32	0.04
5	95.44	0.10	3.22	1.07	0.17	0.02

Table 5.5: Residual functional variability for first 5 dimensions ($d = 128$)

Using the residual functional variability as a criterion, we make the following remarks:

- The STD method is again the most inferior, as supported by the largest residual variability.
- While PCA is the most effective for low dimensional case $d = 16$, it is interesting to note that an appropriate choice of the direction for the PC method can surpass the effectiveness of PCA, as demonstrated in the high-dimensional case $d = 128$.

5.1.3 Variance Reduction Ratios

The comparisons in the last two subsections focus only on the decomposition matrices $\boldsymbol{A}$. Ultimately we are interested in the performance of the QMC-based PGM involving these matrices. For each PGM method, scrambled Sobol sequence of $N = 2048$ and $N = 4096$ points are used to estimate the Asian option. This procedure is replicated $M = 100$ times using independent scrambled Sobol sequence to produce an estimate of the variance of the QMC-based option estimator. For each QMC-based PGM method, we compute the variance reduction ratio, which is the ratio of the variance of MC estimator to the QMC estimator. The results are reported in Tables 5.1.3 and 5.1.3 for $K \in \{90, 100, 110\}$.

d	K	STD	PCA	BB	NBB	DFT	PC	MIXC
16	90	322.45	6484.35	2684.55	3394.16	4853.56	11882.72	5269.47
16	100	82.29	6412.33	1027.68	2342.00	3099.81	4978.82	4817.52
16	110	27.23	2713.99	496.45	715.27	986.59	2115.94	2244.94
64	90	142.79	8024.80	1940.91	3612.66	5818.09	5945.63	7555.55
64	100	30.65	4823.20	627.13	2090.61	3202.38	5254.88	3655.76
64	110	11.10	2668.20	266.07	771.63	2100.27	3012.09	1756.94

Table 5.6: Variance reduction ratio for arithmetic Asian option with $M = 100, N = 2048$

d	K	STD	PCA	BB	NBB	DFT	PC	MIXC
16	90	362.04	18947.21	3366.17	7463.02	9088.63	29688.88	15261.10
16	100	72.67	14311.61	1589.59	2529.06	5816.14	9771.25	9442.65
16	110	26.67	5339.41	605.96	1201.60	1655.87	6412.46	3177.60
64	90	114.64	19571.17	3552.78	6408.98	10871.71	16300.45	14394.37
64	100	31.20	13929.32	1458.89	3140.33	8160.25	13932.35	10450.17
64	110	15.86	4248.61	624.94	1063.84	2815.53	4612.31	3731.53

Table 5.7: Variance reduction ratio for arithmetic Asian option with $M = 100, N = 4096$

Note that for our proposed DC method, we consider two implementations. The first method, which is labeled as PC, is simulated using the decomposition matrix $\boldsymbol{A}$ obtained from Algorithm 4.2.1. The second method, which is labeled as MIXC, is simulated according to Algorithm 4.2.4 where the importance index is defined as the rank of $(var(f_1), ..., var(f_d))$, and the number of important variables (i.e.k_{MIXC} in Algorithm 4.2.4) is 1 if $d = 16$ and 2 if $d = 64$.

Based on the simulated results, we make the following remarks:

- The inefficiency of the STD method is clearly highlighted. In all cases, the variance reduction ratio of the STD method is much smaller compared to other PGMs. This is consistent with performance we saw in the last two subsections.

- Comparing between BB and NBB, the examples support the notion that there is an advantage in an optimal selection of the sampling points. In all cases, NBB is more effective than the BB.
- On the other hand, the remaining PGMs; i.e. PCA, DFT, PC and MIXC, outperform NBB. However, none of these methods is uniformly best.
- It is of interest to note that it is possible for our proposed PC and MIXC to outperform the other PGMs.

5.2 Weighted Arithmetic Average Options

The second example we will study is the discrete weighted average call option with its payoff function at maturity T given by

$$h(\boldsymbol{S}) = \max\left(\sum_{i=1}^{d} w_i S_i, K\right).$$

By setting $f(\boldsymbol{x}) = f_1(x_1) + \cdots f_d(x_d)$, where $f_i = w_i \exp(\mu_i + \sigma x_i)$, then the above payoff function can be written in term of $g(\boldsymbol{x})$ as

$$g(\boldsymbol{x}) = \max(f(\boldsymbol{x}) - K, 0) = \max(f_1(x_1) + ... + f_d(x_d) - K, 0)$$

The base case parameters and comparison methods are the same as Section 5.1. We similarly use the same three criteria as in the last subsection to evaluate the relative efficiency of the various QMC-based PGMs.

5.2.1 Explained Variability

We first examine the explained variability. Tables 5.2.1 and 5.2.1 show the residual functional variability for $d = 16$, and $d = 64$. In Table 5.2.1, the weights for the weighted

average option are generated randomly using Matlab function $rand(d, 1)$ times a normalized constant c for $d = 16$. i.e weights $= c$ (0.0545 0.0089 0.0722 0.1047 0.0267 0.0628 0.0633 0.1154 0.0739 0.0345 0.0638 0.1181 0.0475 0.0777 0.0090 0.0670)^T, for a normalized constant c

Component	STD	PCA	BB	NBB	DCT	PC
1	11.76	81.19	68.75	75.12	80.37	80.74
2	22.79	90.27	84.56	84.07	89.41	89.55
3	33.09	93.58	88.60	89.71	92.75	92.64
4	42.65	95.29	89.71	92.03	94.51	94.49
5	51.47	96.36	90.07	94.36	95.63	95.50

Table 5.8: Explained variability for first 5 dimensions ($d = 16$)

For the results in Table 5.2.2, the same method is used to randomly generate the required weights for $d = 64$, i.e weights $= c$ (0.3167 0.9243 0.6706 0.4378 0.4319 0.8031 0.9218 0.7446 0.3153 0.2718 0.0708 0.0306 0.5531 0.5124 0.9137 0.6507 0.0622 0.8917 0.4113 0.1775 0.3055 0.2941 0.1427 0.7375 0.1882 0.1713 0.2435 0.1012 0.1678 0.6196 0.0993 0.7870 0.9192 0.2590 0.8547 0.6581 0.4024 0.3828 0.3297 0.9043 0.2946 0.2916 0.6413 0.1349 0.9629 0.2118 0.0036 0.8760 0.2147 0.4362 0.0545 0.2252 0.0452 0.4036 0.5068 0.9441 0.9390 0.9031 0.4026 0.7579 0.9583 0.9830 0.2311 0.3822)^T, for a normalized constant c.

Component	STD	PCA	BB	NBB	DCT	PC
1	3.08	81.07	67.19	75.01	80.85	80.28
2	6.11	90.08	83.61	84.25	89.84	89.49
3	9.09	93.32	87.72	89.16	93.08	92.93
4	12.02	94.98	88.75	91.47	94.74	94.55
5	14.90	95.99	89.01	93.79	95.75	95.47

Table 5.9: Explained variability for first 5 dimensions ($d = 64$)

Using the explained variability as a criterion, we make the following remarks:

- The STD method is the worst for both cases.
- In term of the explained variability, other methods such as BB, NBB, DCT and PC are inferior to PCA even if we change the structure of f dramatically, although they are rather close to the optimal values of PCA.

5.2.2 Residual Functional variability

Then, we examine the residual functional variability. Tables 5.2.2 and 5.2.2 show the residual functional variability for $d = 16$, and $d = 64$. In Table 5.2.2, the weights are defined to be the same as Section 5.2.1 for $d = 16$.

Component	STD	PCA	BB	NBB	DCT	PC
1	84.08	1.61	24.41	10.75	3.95	1.04
2	69.78	0.13	5.18	3.59	1.67	0.30
3	55.75	0.13	3.36	2.54	1.51	0.08
4	43.73	0.10	3.29	1.85	1.28	0.08
5	34.37	0.10	3.26	0.73	1.18	0.07

Table 5.10: Residual functional variability for first 5 dimensions ($d = 16$)

For the results in Table 5.2.2, the weights are defined to be the same as Section 5.2.1 for $d = 64$.

Component	STD	PCA	BB	NBB	DCT	PC
1	95.53	0.83	20.63	11.03	1.49	2.40
2	91.15	0.46	5.41	5.28	1.05	0.34
3	87.02	0.20	3.65	2.71	0.72	0.08
4	83.07	0.17	3.34	1.86	0.67	0.05
5	79.24	0.06	3.29	0.85	0.49	0.03

Table 5.11: Residual functional variability for first 5 dimensions ($d = 64$)

Using the residual functional variability as a criterion, we make the following remarks:

- The STD method is the worst for both cases.
- While PCA is still effective, it is interesting to note that an appropriate choice of the direction for the PC method can significantly surpass the effectiveness of PCA, as demonstrated in the high dimensional case $d = 64$.

We can see clearly from the table that the advantage of PCA is not consistent in the residual functional variability. If the structure of f_i's is not proportional to the structure of x_i's, for $i = 1, ..., d$, our projection control method could outperform other methods significantly in the residual functional variability even when the dimension of the function f is not high.

5.2.3 Reduction Factor

The same set of scrambled Sobol sequence is use to estimate the prices of the various weighted average options. In particular we consider two sets of weights prescribed by $w_i = ci^2$ and $w_i = ci^{-2}$ for $i = 1, ..., d$, where c is the normalizing constant so that the sum of the weights is one. The results are reported in Tables 5.2.3 and 5.2.3 for increasing

weights with $K \in \{90, 100, 110\}$, and in Tables 5.2.3 and 5.2.3 for decreasing weights with $K \in \{90, 100, 110\}$. For the MIXC method, we similarly assume that $k_{MIXC} = 1$ for $d = 16$ and $k_{MIXC} = 2$ for $d = 64$ by SC method, and the remaining dimensions are determined by the PC method.

d	K	STD	PCA	BB	NBB	DFT	PC	MIXC
16	90	225.92	7180.99	2583.65	3311.72	6144.58	6922.61	6052.96
16	100	52.26	7146.25	2000.61	2294.21	4853.51	6123.63	6126.26
16	110	28.31	2988.92	867.73	1826.61	2473.61	2742.86	2714.79
64	90	43.87	7391.13	2023.04	4807.36	6959.98	6511.40	6946.99
64	100	22.14	7821.84	775.37	2416.56	6210.18	6817.34	6199.42
64	110	7.12	4056.22	362.92	1547.66	3065.60	3557.92	2714.47

Table 5.12: Variance reduction ratio for the weighted average option with increasing weights, $w_i = ci^2$, $M = 100$, $N = 2048$

d	K	STD	PCA	BB	NBB	DFT	PC	MIXC
16	90	272.66	14845.65	4800.49	6819.55	13837.39	14258.88	12039.65
16	100	58.02	9981.90	2182.83	3856.72	6596.42	9484.97	8520.11
16	110	26.48	7236.43	1203.44	2823.29	4523.19	6535.86	4817.49
64	90	76.16	13239.05	4177.72	6013.64	12093.82	13033.42	10294.62
64	100	26.03	8748.33	2302.03	3542.41	7219.37	7571.69	7976.75
64	110	11.51	6299.22	1212.06	2463.81	3672.95	5401.33	4541.08

Table 5.13: Variance reduction ratio for the weighted average option with increasing weights, $w_i = ci^2$, $M = 100$, $N = 4096$

(2) We consider the case that $w_i = ci^{-2}$ for $i = 1, ..., d$, where c is a normalized constant.

d	K	STD	PCA	BB	NBB	DFT	PC	MIXC
16	90	7433.90	1631.86	1885.07	1497.32	859.01	11159.20	8230.24
16	100	784.21	86.07	56.53	259.72	40.70	3556.65	2538.59
16	110	74.26	14.80	11.90	21.28	9.16	165.50	138.26
64	90	10299.45	9911.55	13178.39	11469.12	6565.57	12382.78	12344.59
64	100	359.40	36.00	88.13	36.22	16.43	1180.41	936.69
64	110	2.33	1.47	1.64	1.23	1.57	2.37	1.47

Table 5.14: Variance reduction ratio for the weighted average option with decreasing weights, $w_i = ci^{-2}$, $M = 100$, $N = 2048$

d	K	STD	PCA	BB	NBB	DFT	PC	MIXC
16	90	11784.23	1868.37	2995.59	2652.23	977.74	17032.18	21112.27
16	100	1404.74	110.34	236.29	353.96	43.73	3295.23	4145.52
16	110	101.93	15.74	20.64	28.74	6.37	229.35	285.19
64	90	33312.80	20276.39	23767.65	26630.11	7485.13	34679.79	32546.59
64	100	394.26	28.17	111.61	57.42	12.98	1445.53	1402.10
64	110	1.94	0.87	1.51	1.53	0.91	2.34	2.71

Table 5.15: Variance reduction ratio for the weighted average option with decreasing weights, $w_i = ci^{-2}$, $M = 100$, $N = 4096$

Based on the simulated results, we make the following remarks:

- For the average option with increasing weights, the relative efficiencies of the PGMs are consistent with the Asian call option that was discussed in Section 5.1.
- On the other, for the weights that are decreasing, interestingly the STD method outperforms the other PGMs except our proposed PC and MIXC methods. These results indicate that by only comparing the explained and residual variability is not sufficient to assess the quality of the PGM for simulating the option prices.

- The above remark also fosters the notion that the structure of the payoff functions is important in designing a good decomposition matrix $\boldsymbol{A}$. Methods that do not take into consideration the structure of the payoff function may have highly variable relative performance. For example let us consider the increasing weight average option with $d = 16, N = 4096$ and $K = 100$. The PCA construction yields an impressive variance reduction ratio of 9982, which compares favourably to the STD method that increases the efficiency by only 58 times. In contrast, if the same methods were used to simulate the average option with weights are decreasing, the relative performance of these methods change dramatically. In this case, PCA is only 110 times more efficiency while the STD method attains a remarkable order of efficiency of 1405 times.

- For methods that do take into account the payoff function of the option, the results are consistently competitive, although the methods may not uniformly outperform the other PGMs. For the same increasing and decreasing average options discussed above, the corresponding variance reduction ratios are 9485 and 3295 for the PC method while 8520 and 4146 for the MIXC method.

- The sensitivity of the PGMs that do not take into consideration the payoff function is also highlighted among the methods of PCA, BB, NBB and DCT. For example, PCA and DFT outperform BB and NBB for increasing weights while the relative perform is reversed when the weights are decreasing.

Here, we can see clearly that if the variability of f_i's are not proportional to the variability of x_i's. i.e. if the structure of $\boldsymbol{Y}$ is different from the structure of $\boldsymbol{x}$, it is significant to apply our DC methods. On the other hand, PCA method is not always to be the optimal solution in the weighted Asian option, due to the importance of prices at some specific times if we change the weights significantly.

5.3 Weighted Geometric Average Options

The third example we will study is the discrete geometric weighted average call option with its payoff function given by

$$h(\boldsymbol{S}) = \max\left(\exp\left(\sum_{i=1}^{d} w_i \log(S_i)\right), K\right).$$

By setting $f(\boldsymbol{x}) = \psi(f_1(x_1), \cdots, f_d(x_d))$, where $f_i = w_i(\mu_i + \sigma x_i)$ and $\psi(f_1(x_1), \cdots, f_d(x_d))$ $= \max(\exp(f_1(x_1) + ... + f_d(x_d)), K,)$, then the above payoff function can be written in term of $g(\boldsymbol{x})$ as

$$g(\boldsymbol{x}) = \max(f(\boldsymbol{x}) - K, 0) = \max(\psi(f_1(x_1), \cdots, f_d(x_d)) - K, 0)$$

The base case parameters are the same as Section 5.1. In addition to the methods we considered in the last subsection, we also consider the Orthogonal transform (OT) of Wang and Tan (2012) with $\boldsymbol{A_0} = \boldsymbol{A_{pca}}$ and $\boldsymbol{q}$ = weight (i.e. $\boldsymbol{w}$); see Section 4.1.4. i.e. We let the first column $\boldsymbol{U_1}$ of OT method is given by

$$\boldsymbol{U_1} = \frac{1}{D}\boldsymbol{A_{pca}}^T\boldsymbol{q} \tag{5.4}$$

where $\boldsymbol{D} = \sqrt{\boldsymbol{q}^T\boldsymbol{\Sigma}\boldsymbol{q}}$, corresponding to the vector $\boldsymbol{q} = \boldsymbol{w}$. The remaining columns are constructed by Gram-Schmidt process.

Furthermore, besides the PC method according to Algorithm 4.2.1, the method labelled as PC2 is another implementation of PC method as motivated by the OT method of Wang and Tan (2012). In this case, the first column direction is controlled by

$$\boldsymbol{v_1} = \frac{1}{\boldsymbol{D}}(\boldsymbol{J^{-1}}\boldsymbol{A_{f_{pca}}})^T\boldsymbol{q}. \tag{5.5}$$

where $\boldsymbol{A_{f_{pca}}}$ is the PCA construction of $\boldsymbol{\Sigma}_f$ defined in Section 4.1.2. The rest of dimensions are constructed by the PC method according to the Algorithm 4.2.2. The third DC method, which is labeled as MIXC, is simulated according to Algorithm 4.2.4 where the importance

index is defined as the rank of $(var(f_1), ..., var(f_d))$, and the number of important variables (i.e.k_{MIXC} in Algorithm 4.2.4) is 1 for both $d = 16$ and $d = 64$.

The results of the variance reduction ratio are reported in Tables 5.3, 5.3for increasing weights with $K \in \{90, 100, 110\}$, and in Tables 5.3, 5.3 for decreasing weights with $K \in \{90, 100, 110\}$.

(1) We consider the case that $w_i = ci^2$ for $i = 1, ..., d$, where c is a normalized constant.

d	K	STD	PCA	BB	NBB	DFT	OT	PC	MIXC	PC2
16	90	186.9	6009.8	4257.4	3975.7	4715.8	6307.0	6100.1	5811.4	6307.0
16	100	55.2	5704.4	1905.1	2942.2	4136.4	5991.1	5282.3	4523.7	5991.1
16	110	31.1	2706.1	881.7	1726.8	1888.8	3008.5	2567.5	2356.3	3008.5
64	90	51.8	7247.0	2206.9	5333.0	6786.0	7604.8	7024.7	6057.7	7604.8
64	100	17.9	4329.5	985.2	1866.6	4154.9	4441.2	4009.5	4513.9	4441.2
64	110	8.4	2916.0	408.3	1147.4	2381.0	3327.0	3199.3	2776.9	3327.0

Table 5.16: Variance reduction ratio for the geometric weighted Average option with increasing weights, $M = 100$, $N = 2048$

d	K	STD	PCA	BB	NBB	DFT
16	90	232.0	16425.5	5416.8	6768.4	12349.6
16	100	65.0	6754.4	2042.0	3047.4	5145.1
16	110	25.7	6405.9	1086.1	2012.1	3439.9
64	90	66.5	21448.6	3880.3	6146.9	15506.9
64	100	25.0	13426.9	2598.0	4605.8	9342.5
64	110	12.6	6417.0	859.7	2616.3	4659.5

d	K	OT	PC	MIXC	PC2
16	90	17498.9	15905.5	13824.6	17498.9
16	100	7270.3	6642.9	6312.6	7270.3
16	110	6533.8	6105.6	5047.8	6533.8
64	90	21863.3	21573.5	19194.1	21863.3
64	100	14820.6	12714.1	9950.3	14820.6
64	110	7055.4	5710.1	6234.1	7055.4

Table 5.17: Variance reduction ratio for the geometric weighted Average option with increasing weights, $M = 100$, $N = 4096$

(2) We consider the case that $w_i = ci^{-2}$ where c is a normalized constant.

d	K	STD	PCA	BB	NBB	DFT
16	90	4651.5	1329.0	1169.2	1889.8	989.7
16	100	806.3	83.1	54.0	219.8	54.7
16	110	61.8	16.3	11.7	21.8	6.5
64	90	17100.1	13011.6	14152.9	12405.9	7882.4
64	100	354.6	36.9	75.7	55.1	18.6
64	110	3.1	1.9	3.2	2.1	2.5

d	K	OT	PC	MIXC	PC2
16	90	13960.2	10417.6	8030.1	13960.2
16	100	8475.3	2318.9	2391.1	8475.3
16	110	530.1	156.6	176.6	530.1
64	90	19018.1	15820.2	12281.6	19018.1
64	100	8475.5	927.8	1036.7	8475.5
64	110	4.7	5.1	2.8	4.7

Table 5.18: Variance reduction ratio for the geometric weighted Asian option with decreasing weights, $M = 100$, $N = 2048$

d	K	STD	PCA	BB	NBB	DFT
16	90	8557.2	1885.7	3540.8	2867.9	630.2
16	100	1118.9	87.5	255.2	375.3	39.6
16	110	110.9	11.2	25.9	37.0	6.6
64	90	37028.1	28412.4	30638.3	25009.1	11489.4
64	100	493.8	30.1	83.6	46.5	15.5
64	110	1.6	1.1	1.2	1.1	0.9
d	K	OT	PC	MIXC	PC2	
16	90	55989.6	16191.1	17516.8	55989.6	
16	100	16217.9	3368.8	3918.6	16217.9	
16	110	1402.1	345.5	240.7	1402.1	
64	90	39337.3	28923.3	33287.1	39337.3	
64	100	21652.5	1828.7	1324.8	21652.5	
64	110	5.6	3.5	2.7	5.6	

Table 5.19: Variance reduction ratio for the geometric weighted Asian option with decreasing weights, $M = 100$, $N = 4096$

Based on the simulated results, we make the following remarks:

- Except the OT and PC2 methods, which control the first direction of the decomposition matrix, our PC and MIXC methods can outperform the other methods in both cases. It highlights that considering the functional structure is very important to improve the effectiveness of the PGMs.
- For the methods that do not consider the functional structures, we can see that in the case of increasing weights, PCA outperforms the others and in the case of decreasing weights, STD outperforms the others. However, none of these methods is uniformly the best.
- Comparing PC and MIXC, we see that our PC method, which is the same as the LT method in this case, performs well in the case that f_i's are linear combinations of x_i's.

However, MIXC can still outperform PC in some cases. This also highlights that the importance index function may take some effects on the effectiveness of PGMs.

- OT and PC2 methods outperform all the other methods, which means that controlling the first direction is very important in the weighted geometric average option.
- It is of interest to note that it is possible for our proposed PC method to perform exactly the same as the optimal decomposition transformed by the OT method.

5.4 Lookback Options

In this section, we will study a type of exotic option known as lookback options. In particular, we consider a discrete fixed-strike weighted lookback call option with the following payoff structure

$$h(\boldsymbol{S}) = \max\left(\max\left(S_1, S_2..., S_d\right), K\right).$$

By setting $f_i = \exp(\mu_i + \sigma x_i)$ and $\psi(f_1(x_1), \cdots, f_d(x_d)) = \max\left(f_1(x_1), ..., f_d(x_d)\right)$, then the above payoff function can be written in term of $g(\boldsymbol{x})$ as

$$g(\boldsymbol{x}) = \max(f(\boldsymbol{x}) - K, 0) = \max(\psi(f_1(x_1), \cdots, f_d(x_d)) - K, 0)$$

The base case parameters and comparison methods are the same as Section 5.1. The results of comparing the variance reduction ratio are reported in Tables 5.4 and 5.4, for $\sigma = 0.2$ and $K \in \{90, 100, 110\}$, and in Tables 5.4 and 5.4, for $\sigma = 0.3$ and $K \in \{90, 100, 110\}$.

d	K	STD	PCA	BB	NBB	DCT	PC	MIXC
16	90	87.47	147.68	242.67	228.98	145.10	165.80	248.19
16	100	95.93	139.37	204.84	212.30	155.07	107.46	183.41
16	110	39.14	99.85	193.25	238.52	78.35	79.59	198.08
64	90	10.18	137.30	112.18	95.97	151.84	100.09	156.14
64	100	12.08	109.69	107.36	110.71	97.38	128.18	183.32
64	110	10.06	87.55	95.77	89.78	95.39	102.51	123.84

Table 5.20: Variance reduction ratio for Lookback option with $M = 100$, $N = 2048$, $\sigma = 0.2$

d	K	STD	PCA	BB	NBB	DCT	PC	MIXC
16	90	96.86	145.57	287.36	265.44	115.26	89.87	247.50
16	100	82.71	112.21	255.04	245.71	140.09	85.53	272.69
16	110	62.39	64.86	297.58	278.34	76.10	56.99	289.13
64	90	18.12	82.92	169.18	75.72	91.31	114.97	153.29
64	100	11.19	100.87	189.64	68.42	108.56	100.39	133.54
64	110	12.27	84.67	132.60	80.36	69.74	73.64	145.79

Table 5.21: Variance reduction ratio for Lookback option with $M = 100$, $N = 4096$, $\sigma = 0.2$

d	K	STD	PCA	BB	NBB	DCT	PC	MIXC
16	90	66.87	92.87	221.89	226.32	103.60	75.36	253.91
16	100	55.69	98.57	196.06	196.48	116.67	94.35	194.63
16	110	45.53	81.55	252.23	190.42	73.38	68.32	165.77
64	90	10.00	99.70	114.29	107.87	100.95	90.90	127.20
64	100	11.59	82.91	127.59	102.23	96.70	88.74	174.31
64	110	9.20	88.34	90.54	75.39	77.72	77.15	136.20

Table 5.22: Variance reduction ratio for Lookback option with $M = 100$, $N = 2048$, $\sigma = 0.3$

d	K	STD	PCA	BB	NBB	DCT	PC	MIXC
16	90	66.19	85.05	230.64	283.07	90.39	68.41	210.46
16	100	78.08	62.37	233.47	322.05	82.68	62.09	263.73
16	110	47.37	54.29	233.14	248.62	59.74	49.66	180.18
64	90	11.15	69.66	152.84	107.28	72.79	69.97	170.17
64	100	12.16	61.39	176.19	92.48	65.87	69.93	138.61
64	110	11.19	65.28	164.50	84.90	54.99	49.71	144.05

Table 5.23: Variance reduction ratio for Lookback option with $M = 100$, $N = 4096$, $\sigma = 0.3$

Note that for our proposed DC method, we consider three implementations. The first method, which is labelled as PC, is simulated using the decomposition matrix $\boldsymbol{A}$ obtained from Algorithm 4.2.1. The second method, which is labeled as SC, is simulated according to Algorithm 4.2.4 where the importance index is defined as the rank of $(E(f_1), ..., E(f_d))$, and the number of important variables (i.e.k_{MIXC} in Algorithm 4.2.4) is d for $d = 16$ and 1 for $d = 64$. In other words, MIXC becomes SC for $d = 16$.

Based on the simulated results, we make the following remarks:

- BB, NBB and MIXC perform better than the other methods. However, among these three methods, there's no method which has consistent advantage.
- It is of interest to note that in this situation, determine the importance of functions is more important than treating all functions equivalently. In other words, all the methods that controlling the column directions of decomposition matrix performs badly in the Lookback option.

5.5 Digital Options

In this section, we will study the digital options. In particular, we consider the following payoff structure

$$\begin{aligned}
h(\boldsymbol{S}) &= \frac{1}{d}\sum_{i=1}^{d}(S_i - S_{i-1})_+^0 S_i \\
&= \frac{1}{d}\sum_{i=1}^{d}(S_i - S_{i-1})_+^0 (S_i - S_{i-1} + S_{i-1}) \\
&= \frac{1}{d}\sum_{i=1}^{d}(S_i - S_{i-1})_+^0 (S_i - S_{i-1}) + (S_i - S_{i-1})_+^0 S_{i-1} \\
&= \frac{1}{d}\left[\left(\sum_{i=1}^{d}(S_i - S_{i-1})_+^0\right) S_0 + \left(\sum_{i=1}^{d}(S_i - S_{i-1})_+^0\right)(S_1 - S_0)\right. \\
&+ \left.\left(\sum_{i=2}^{d}(S_i - S_{i-1})_+^0\right)(S_2 - S_1)\right. \\
&+ \left.\cdots + \left(\sum_{i=d}^{d}(S_i - S_{i-1})_+^0\right)(S_d - S_{d-1})\right] \\
&= \frac{1}{d}\left[w_1 S_0 + w_1(S_1 - S_0) + w_2(S_2 - S_1) + \cdots + w_d(S_d - S_{d-1})\right]
\end{aligned}$$

where $(x)_+^0$ is equal to 1 if $x > 0$ and is 0 otherwise, for $x \in R$, and $\boldsymbol{w}$ is a weight vector of $S_i - S_{i-1}$'s. i.e. $w_k = \sum_{i=k}^{d}(S_i - S_{i-1})_+^0$ for $i = 1, ..., d$,$k = 1, ..., d$. For $d > 1$, it can be viewed

as a portfolio of digital options, or as a ratchet option. Ratchet options are useful to hedge equity-linked index annuities. Papageorgiou (2002) showed that both BB and PCA are worse than STD. If the optimal solution is to make $(S_i - S_{i-1})_+^0$'s evaluate to be one, then the weight vector becomes $(d, d-1, ..., 1)^T$. i.e. the importance of $S_i - S_{i-1}$ are decreasing for $i = 1, .., d$. Conditional on $(S_i - S_{i-1})_+^0 = 1$, for $i = 1, ..., d$, by setting $f_1 = w_1(S_1 - S_0)$, $f_2 = w_2(S_2 - S_1)$, ..., $f_d = w_d(S_d - S_{d-1})$. Then the above payoff function can be written in term of $g(\boldsymbol{x})$ as

$$f(\boldsymbol{x}) = c + f_1(\boldsymbol{x}) + \cdots + f_d(\boldsymbol{x}),$$

where c is a constant.

The base case parameters and comparison methods are the same as Section 5.1. For each PGM method, scrambled Sobol sequence of $N = 2048$ points are used to estimate the digital option. This procedure is replicated $M = 100$ times using independent scrambled Sobol sequence to produce an estimate of the variance of the QMC-based option estimator. The results of comparing the variance reduction ratio are reported in Tables 5.5 and 5.5.

Component	Mean	S.E	Reduction Factor
STD:	59.035801	0.002469	287.698542
PCA:	59.047774	0.017072	6.015416
BB:	59.056150	0.015955	6.887032
NBB:	59.040820	0.011499	13.260198
DCT:	59.010077	0.015903	6.931890
SC:	59.035801	0.002469	287.698542

Table 5.24: Variance reduction ratio for digital option with $d = 16$, $M = 100$, $N = 2048$

Component	Mean	S.E	Reduction Factor
STD:	55.772191	0.002683	100.427258
PCA:	55.767075	0.009370	8.232068
BB:	55.758646	0.007739	12.067155
NBB:	55.777251	0.010416	6.661215
DCT:	55.773437	0.008542	9.904400
MIXC:	55.772191	0.002683	100.427258

Table 5.25: Variance reduction ratio for digital option with $d = 64$, $M = 100$, $N = 2048$

Note that for our proposed DC method, we consider only one implementation, which is labeled as SC, is simulated according to Algorithm 4.2.3 where the importance index is defined as the rank of $(E(f_1), ..., E(f_d))$.

Based on the simulated results, we make the following remarks:

- STD and SC outperform all the other methods. The decreasing order of generating Brownian paths is the most important way in the digital option.
- None of the methods PCA, BB, NBB and DCT has a consistent advantage over each other.
- It is of interest to note that our SC method focusing on the difference of the prices between time t_i and t_{i-1}, for $i = 1, ..., d$, performs exactly the same as STD.
- Due to the fact that f_i contains the term $(S_i - S_{i-1})_+^0$ for $i = 1, ..., d$, it is hard to calculate the Jacobian matrix in our additive structure of f. So it is hard to apply our PC method in this situation.

Now let us make an assumption that the STD construction, denoted as $\boldsymbol{A_{std}}$,is the best decomposition of $\boldsymbol{\Sigma}$ without considering the structure of the function f. We assume that

our new f_i's are defined as

$$\begin{aligned} f_1 &= \frac{1}{d}(S_1 - S_0)_+^0 S_1, \\ f_2 &= \frac{1}{d}(S_2 - S_1)_+^0 S_2 \\ \cdots & \\ f_d &= \frac{1}{d}(S_d - S_{d-1})_+^0 S_d. \end{aligned} \tag{5.6}$$

Then, the above payoff function can be written in term of $f(\boldsymbol{x})$ as

$$f(\boldsymbol{x}) = f_1(\boldsymbol{x}) + \cdots + f_d(\boldsymbol{x}).$$

According to the formula from Papageorgiou (2001), we have

$$E[(S_j - S_{j-1})_+^0 (S_i - S_{i-1})_+^0 S_j S_i]$$

$$= S_0^2 \Phi(\beta(r,\sigma,h) + \sigma\sqrt{h})\Phi(\beta(r,\sigma,h) + 2\sigma\sqrt{h}) \exp(r(i+j)h) \exp(\sigma^2 ih)$$

and

$$E[(S_j - S_{j-1})_+^0 S_j^2] = S_0^2 \Phi(\beta(r,\sigma,h) + 2\sigma\sqrt{h}) \exp(2rjh) \exp(\sigma^2 jh)$$

where $i, j \in \{1/d, 2/d..., 1\}$. So, we can calculate the covariance matrix $\boldsymbol{\Sigma}_f$, and its corresponding PCA decomposition, denoted as $\boldsymbol{A_f}$. Then, we get the Jacobian matrix $\boldsymbol{J}$ by $\boldsymbol{J} = \boldsymbol{A_f}/\boldsymbol{A_{std}}$, or equivalently $\boldsymbol{J} = \boldsymbol{A_f}\boldsymbol{A_{std}}^{-1}$. Then we apply the algorithm 4.2.1 to obtain the results reported in Tables 5.5 and 5.5.

d	STD	PCA	BB	NBB	DFT	PC
16	227.05	6.50	7.91	12.83	6.64	271.59
64	79.78	11.98	11.28	6.00	8.05	79.75

Table 5.26: Variance reduction ratio for digital option with $M = 100$, $N = 2048$

d	STD	PCA	BB	NBB	DFT	PC
16	339.46	7.69	9.56	15.97	7.36	354.70
64	105.50	10.57	11.40	4.51	8.74	113.97

Table 5.27: Variance reduction ratio for digital option with $M = 100$, $N = 4096$

Note that for our proposed DC method, we consider only one implementation, which is labeled as PC, is simulated according to Algorithm 4.2.1 with the Jacobian matrix calculated above.

Based on the simulated results, we make the following remarks:

- Our PC method can outperform the STD method, but not always. See Table 5.5 with $d = 64$.
- For PCA, BB, NBB and DCT, no one has a consistent advantage among these four methods.
- It is of interest to note that our PC method can perform better by calculating the functional covariance matrix directly in the case that it is hard to do partial derivatives for calculating the Jacobian matrix. In other words, in the situation that it's not easy to calculate the Jacobian matrix, we can calculate the covariance matrix of f_i's directly, and combine a prior decomposition to get a new adjusted decomposition by the PC method.

5.6 Asian Basket Options

The next example we take a look is the most popular multi-asset option, which is called the Basket option. Assuming a multi-dimensional BS market with m securities and one risk-free asset, we let $\boldsymbol{B(t)} = (B_1(t), ..., B_m(t))$ be an m-dimensional Brownian motion with correlated components. Moreover, let ρ_{ik} be the constant instantaneous correlation between $B_i(t)$ and $B_k(t)$, $S_i(t)$ be the i^{th} asset price at time t, $\sigma_i(t)$ be the instantaneous

time-dependent volatility of the i^{th} asset and r be the continuously compounded risk-free measure.

In the risk-neutral probability, the dynamics of the assets are defined as

$$dS_i(t) = rS_i(t)dt + \sigma_i(t)S_i(t)dB_i(t) \tag{5.7}$$

for $i = 1, ..., m$. The solution of (5.7) is equal to

$$S_i(t) = S_i(0) \exp\left(\int_0^t \left(r - \frac{\sigma_i^2(s)}{2} \right) ds + \int_0^t \sigma_i(s) dB_i(s) \right) \tag{5.8}$$

for $i = 1, ..., m$. Pricing the Asian Basket option requires the discrete averaging of (5.8) at a finite set of times $t_1, ..., t_n$. This sampling procedure gives

$$S_i(t_j) = S_i(0) \exp\left(\int_0^{t_j} \left(r - \frac{\sigma_i^2(t)}{2} \right) dt + Z_i(t_j) \right) \tag{5.9}$$

where $i = 1, ..., m$, $j = 1, ..., n$, and the vector $\boldsymbol{Z} = (Z_1(t_1),\ ...,\ Z_1(t_n),\ ....,\ Z_m(t_1),\ ...,\ Z_m(t_n))^T$ is a m by n normal random variable with mean zero and covariance matrix

$$\boldsymbol{\Sigma}_{mn} = \begin{pmatrix} \boldsymbol{\Sigma}(t_1) & \boldsymbol{\Sigma}(t_1) & ... & \boldsymbol{\Sigma}(t_1) \\ \boldsymbol{\Sigma}(t_1) & \boldsymbol{\Sigma}(t_2) & ... & \boldsymbol{\Sigma}(t_2) \\ ... & ... & ... & ... \\ \boldsymbol{\Sigma}(t_1) & \boldsymbol{\Sigma}(t_2) & ... & \boldsymbol{\Sigma}(t_n) \end{pmatrix},$$

where the elements $(\boldsymbol{\Sigma}(t_j))_{ik} = \int_0^{t_j} \rho_{ik}\sigma_i(t)\sigma_k(t)dt$, for $i = 1, ..., m$, $k = 1, ..., m$, and $j = 1, ..., n$. In the case of constant volatilities, the covariance matrix becomes

$$\boldsymbol{\Sigma}_{mn} = \begin{pmatrix} t_1\boldsymbol{\Sigma} & t_1\boldsymbol{\Sigma} & ... & t_1\boldsymbol{\Sigma} \\ t_1\boldsymbol{\Sigma} & t_2\boldsymbol{\Sigma} & ... & t_2\boldsymbol{\Sigma} \\ ... & ... & ... & ... \\ t_1\boldsymbol{\Sigma} & t_2\boldsymbol{\Sigma} & ... & t_n\boldsymbol{\Sigma} \end{pmatrix}.$$

More details of comparing with Asian Basket options are written by Imai and Tan (2006). Our interest here is to look at some interesting examples using our MIXC methods, compared with the forward and PCA constructions. By setting $f(\boldsymbol{Z}) = f_{11} + ... + f_{1n} + ... + f_{m1} + ... + f_{mn}$, where $f_{ij} = w_{ij}S_i(t_j) = w_{ij}S_i(0)\exp\left[(r - \frac{\sigma_i^2)}{2})t_j + Z_i(t_j)\right]$ for $i = 1, ..., m$ and $j = 1, ..., n$, then the above payoff function can be written in term of $g(\boldsymbol{Z})$ as

$$g(\boldsymbol{Z}) = \max(f(\boldsymbol{Z}) - K, 0).$$

For our MIXC method, we first simulate m different assets by algorithm 4.2.4 according to the SC method by the importance index of $(E(f_{11}), ..., E(f_{1n}), ..., E(f_{m1}), ..., E(f_{mn}))^T$, and build all independent elements in the covariance matrix $\boldsymbol{\Sigma}_{mn}$. Then, we use the Cholesky decomposition of the covariance matrix of σ_i's, denoted by L, multiplied by each m by m subset of $\boldsymbol{\Sigma}_{mn}$. Then, we use the PC method for the rest of dimensions. For our PC method, in order to differentiate from LT method, we use the adjusted method for PCA, i.e. $\boldsymbol{J} = \boldsymbol{A_f}/\boldsymbol{A_{PCA}}$, where $\boldsymbol{A_f}$ is the PCA construction for $\boldsymbol{\Sigma}_f$ and $\boldsymbol{A_{PCA}}$ is the PCA construction for $\boldsymbol{\Sigma}$, so the Jacobian matrix is no longer to be a diagonal matrix, i.e. our PC method is not equivalent to the LT method.

For the parameters, we use $S_i(0) = 100, K \in \{0, 90, 100\}, r = 0.04, T = 1, \sigma_i(0) = 0.1 + \frac{i-1}{9}0.4$, for $i = 1, ..., n$, $\rho \in \{0, 0.4\}$, $M = 10$, $n = 50$, replicationS $m = 10$, and number of points $n = 2048$. The weights are assumed to be equal.

K	STD	PCA	PC	MIXC
90	47.20	548.03	826.56	1957.18
100	2.63	71.40	207.07	55.35

Table 5.28: Reduction Factor for Asian Basket Option with $M = 10$, $n = 50$, $m = 10$, $n = 2048$, $\rho = 0$

K	STD	PCA	PC	MIXC
90	7.67	2453.22	3951.38	1124.26
100	2.02	678.08	2193.05	952.34

Table 5.29: Reduction Factor for Asian Basket Option with $M = 10$, $N = 50$, $m = 10$, $n = 2048$, $\rho = 0.4$

For our proposed DC method, we consider two implementations. The first method, which is labeled as PC, is simulated using the decomposition matrix $\boldsymbol{A}$ obtained from Algorithm 4.2.1. The second method, which is labeled as MIXC, is simulated according to Algorithm 4.2.4 where the importance index is defined as the rank of ($E(f_{11})$, ..., $E(f_{1n})$, ..., $E(f_{m1})$, ..., $E(f_{mn})$), and the number of important variables (i.e.k_{MIXC} in Algorithm 4.2.4) is m.

Based on the simulated results, we make the following remarks:

- Considering the functional variability is more important than the explained variability.
- Note that the PC construction is not always the best decomposition. Some functions may take dominant effects on the total variability.
- Note also that our PC method can outperform the other methods by calculating the functional covariance matrix directly, and combine a prior decomposition to get a better adjusted decomposition.

Hence, considering the normality of the functions is better than maximizing the explained variability, i.e. the PCA method. Thus, our DC methods provides a more general decomposition even in the multiple asset situations.

5.7 Asian Options with Knock-out Feature at Maturity

The payoff function of the Asian option with knock-out feature at the maturity is defined as

$$G = \max(S_A - K, 0) I_{\{S_d \leq H\}}(S), \tag{5.10}$$

where

$$S_A = \sum_{i=1}^{d} w_i S_i \tag{5.11}$$

with $\sum_{i=1}^{d} w_i = 1$, K is the strike price, and H is the barrier. If at the expiration the price of the underlying asset is below the barrier H, then the option becomes the ordinary Asian call option. However, if the price at the expiration is above the barrier H, the option pays zero.

By setting $f(\boldsymbol{x}) = f_1 + \cdots + f_d$, where $f_i = w_i S_i(x_i)$ then the above payoff function can be written in term of $g(\boldsymbol{x})$ as

$$g(\boldsymbol{x}) = \max(f(\boldsymbol{x}) - K, 0) I_{\{S_d \leq H\}}(S).$$

The base case parameters are the same as Section 5.1. The results of comparing the variance reduction ratio are reported in Tables 5.7 and 5.7 with equal weights, $K = 100$, $H \in \{120, 140, 160\}$, and in Tables 5.7 and 5.7 with random weights, $K = 100$, $H \in \{120, 140, 160\}$.

(1) The weights are equal

d	H	STD	PCA	BB	NBB	MIXC
16	120	1.88	5.47	198.48	9.01	581.99
16	140	2.52	10.16	226.19	11.38	599.34
16	160	4.58	20.06	305.89	17.12	494.56
64	120	1.60	5.50	107.91	8.50	454.64
64	140	1.91	8.87	294.48	11.54	458.37
64	160	3.04	14.54	316.31	10.69	523.44

Table 5.30: Variance reduction ratio for Asian options with knock-out feature at maturity with equal weights, $M = 100$, $N = 2048$

d	H	STD	PCA	BB	NBB	MIXC
16	120	1.49	6.58	335.19	12.03	1195.43
16	140	2.87	10.17	633.15	11.29	1868.68
16	160	3.47	19.72	753.17	20.37	1789.28
64	120	1.38	4.54	354.54	11.13	1107.78
64	140	1.21	8.60	486.75	9.59	1722.97
64	160	2.82	19.06	526.79	16.37	1217.77

Table 5.31: Variance reduction ratio for Asian options with knock-out feature at maturity with equal weights, $M = 100$, $N = 4096$

(2) Weight = c rand(d,1), where c is a normalized constant.

d	H	STD	PCA	BB	NBB	MIXC
16	120	1.67	6.72	180.01	6.31	648.91
16	140	3.05	7.64	286.22	10.90	674.60
16	160	3.47	16.46	420.47	16.56	540.42
64	120	1.34	5.42	98.21	7.53	584.55
64	140	1.74	8.80	211.42	8.97	449.80
64	160	3.39	16.42	273.21	10.78	484.49

Table 5.32: Variance reduction ratio for Asian options with knock-out feature at maturity with random weights, $M = 100$, $N = 2048$

d	H	STD	PCA	BB	NBB	MIXC
16	120	1.67	5.70	317.72	8.05	988.47
16	140	2.82	9.40	560.12	9.74	1529.58
16	160	4.82	15.75	703.18	16.73	1434.87
64	120	1.54	5.74	294.46	8.11	881.99
64	140	1.99	7.02	566.33	7.28	2141.41
64	160	2.65	16.16	564.88	15.20	1345.64

Table 5.33: Variance reduction ratio for Asian options with knock-out feature at maturity with random weights, $M = 100$, $N = 4096$

Note that for our proposed DC method, we consider only one implementation, which is labeled as MIXC, is simulated according to Algorithm 4.2.4 with $k_{MIXC} = 1$ and the first Brownian path is generated for the function f_d.

Based on the simulated results, we make the following remarks:

- Both PCA and NBB are only marginally better than STD.
- Our MIXC method can significantly outperform all the other methods in all the situations. The BB construction, on other other hand, is the next most efficient method.

- This suggests that considering the functional structure is very important in the Asian options with knock-out feature.

5.8 Asian options with a Modified Knock-out Feature

Let $\boldsymbol{V_1}$ be the first column of the generating matrix corresponding to PCA. Let $\boldsymbol{V} = b\boldsymbol{V_1}$ $= (v_1, ..., v_d)^T$, which satisfies $\sum_{i=1}^{d} v_i = 1$. The payoff function of the option is defined as

$$G = \max(S_A - K, 0) I_{\bar{S} \leq H}(S) \tag{5.12}$$

with $\bar{S} = \prod_{i=1}^{d} S_i^{v_i}$. i.e. $\bar{S}$ is a specific weighted geometric average of asset prices with weights v_i's.

By setting $f(\boldsymbol{x}) = f_1 + \cdots + f_d$, where $f_i = w_i S_i(x_i)$ then the above payoff function can be written in term of $g(\boldsymbol{x})$ as

$$g(\boldsymbol{x}) = \max(f(\boldsymbol{x}) - K, 0) I_{\bar{S} \leq H}(S).$$

5.8.1 Variance Reduction Ratio

The base case parameters and comparison methods are the same as Section 5.7. The results of comparing the variance reduction ratio are reported in Tables 5.8.1, 5.8.1 and 5.8.1 with equal weights, $K = 100$, $H \in \{120, 140, 160\}$, and in Tables 5.8.1 and 5.8.1 with random weights, $K = 100$, $H \in \{120, 140, 160\}$.

(1) Equally weighted case:

d	H	STD	PCA	BB	NBB	PC
16	120	1.74	675.93	5.62	9.16	681.99
16	140	4.83	500.65	16.01	16.85	521.78
16	160	15.05	376.51	56.79	70.76	369.30
64	120	1.49	861.70	5.16	7.00	926.49
64	140	2.27	547.58	14.85	26.58	621.09
64	160	9.56	483.23	56.24	61.79	492.86

Table 5.34: Variance reduction ratio for Asian options with modified knock-out feature with equal weights, $M = 100$, $N = 2048$

d	H	STD	PCA	BB	NBB	PC
16	120	1.82	8503.10	6.59	6.46	10418.13
16	140	3.93	2020.64	13.54	18.92	2169.43
16	160	19.52	624.24	82.07	67.21	619.70
64	120	1.36	1268.99	6.92	9.84	1427.15
64	140	3.26	1064.71	14.92	21.50	1090.56
64	160	10.84	596.63	55.73	83.75	592.11

Table 5.35: Variance reduction ratio for Asian options with modified knock-out feature with equal weights, $M = 100$, $N = 4096$

(2) Randomly weighted case:

d	H	STD	PCA	BB	NBB	PC
16	120	1.77	499.02	7.07	5.96	668.99
16	140	3.75	519.96	12.13	18.22	544.68
16	160	12.84	372.46	55.52	90.92	386.84
64	120	1.14	902.42	4.26	9.22	950.81
64	140	2.78	581.20	17.28	20.18	647.39
64	160	9.41	392.98	52.16	72.34	399.83

Table 5.36: Variance reduction ratio for Asian options with modified knock-out feature with random weights, $M = 100$, $N = 2048$

d	H	STD	PCA	BB	NBB	PC
16	120	1.26	3125.27	5.78	8.39	3904.63
16	140	3.85	1960.39	12.39	19.48	2254.28
16	160	17.00	623.89	61.83	68.98	624.04
64	120	1.34	1253.17	5.72	8.35	1373.55
64	140	3.43	1202.60	19.54	23.76	1227.50
64	160	10.03	593.28	69.77	80.73	598.25

Table 5.37: Variance reduction ratio for Asian options with modified knock-out feature with random weights, $M = 100$, $N = 4096$

(3) Equal weights and $N = 256$*:*

d	H	STD	PCA	BB	NBB	PC
16	120	1.20	363.94	3.27	6.06	420.24
16	140	3.40	114.30	12.48	12.81	124.73
16	160	11.97	41.03	25.48	32.75	42.91
64	120	1.19	86.36	5.01	5.09	90.15
64	140	3.10	69.05	7.95	13.16	72.23
64	160	5.39	40.24	29.12	30.47	41.17

Table 5.38: Variance reduction ratio for Asian options with modified knock-out feature with equal weights, $M = 100$, $N = 256$

The PC method is simulated according to Algorithm 4.2.2 with the first direction controlled by the first column of PCA. Based on the simulated results, we make the following remarks:

- For this set of examples, PCA construction is now the second most efficient method while our proposed PC method is the most efficient.
- It is of interest to note that our PC method which tends to minimize the residual functional variability can improve the effectiveness in Asian options with modified knock-out feature.
- Note that if the number of paths are small, maximize the functional variability becomes even more important than the explained variability.

5.8.2 Run-time Complexity

Now, we look at the run-time complexity with equal weights, $K = 100$, $H = 120$, and $d \in \{16, 64, 128\}$. The base case parameters and comparison methods are the same as Section 5.7. The unit base of our run-time complexity is in seconds.

(1) Run-time for constructing decomposition matrices (in seconds):

d	STD	PCA	BB	NBB	PC
16	0.00	0.02	0.02	0.02	0.05
64	0.00	0.03	0.02	0.02	0.34
128	0.00	0.03	0.03	0.03	4.40

Table 5.39: Run-time for constructing decomposition matrices with equal weights, $K = 100$, $H = 120$

(2) Run-time for calculating QMC estimator (in seconds):

d	STD	PCA	BB	NBB	PC
16	3.73	3.74	3.70	3.63	3.74
64	14.91	14.73	14.57	14.65	14.41
128	22.12	22.48	21.81	22.35	22.06

Table 5.40: Run-time for calculating QMC estimator with equal weights, $K = 100$, $H = 120$

Based on the simulated results, we make the following remarks:

- In the case of run-time for constructing decomposition matrices, STD takes the lowest run-time, our PC method takes the highest run-time, and all the other methods seem to be close and much faster than our PC method.
- In the case of run-time for calculating the QMC estimator, all methods seem to be close. Our PC method is slightly better than the other methods in higher dimensions.
- The construction run-time complexity of our PC method in higher dimensions is a big issue as we discussed in Section 4.3. The improvement of the run-time for constructing the decomposition matrix by our PC method is still under investigation.

Chapter 6

Conclusions and Further Research

6.1 Conclusions

The purpose of this thesis is to discuss some efficient ways of computing high-dimensional integrals with MC and QMC methods as they recently became the fundamental tools of computer simulation in the area of finance and insurance.

We began with discussing the low discrepancy sequences, which improve the performance of MC simulations because they usually offer shorter computational times and higher accuracy in lower dimensions. As we have seen in Chapter 2, the reason that low discrepancy sequences give a higher accuracy is due to their high uniformity in lower dimensions. We reviewed some efficient algorithms to compute QMC sequences and showed some problems of non-uniformity in higher dimensions. The investigation enables us to better understand the enhanced properties of QMC and shows the reason why QMC can have a better (or no better) performance than MC method for evaluating high-dimensional multivariate integration. The superposition discrepancies of low discrepancy points and random points are almost the same, unless the number of points is huge. Thus, in the situation that using small number of simulation paths, considering the structure of the function is significant in order to maximize the functional variability on the first few dimensions.

In the Chapter 3, we focus on talking about the effective dimensions due to the restricted properties of low discrepancy sequences. We start with the general ANOVA decomposition. We review the notion of effective dimension as truncation dimension and superposition dimension. We see that the truncation dimension and superposition dimension tell us the minimum number of variables and the lowest dimension we need, respectively, in order to keep a certain level of variability of the function. However, we see that computing the truncation dimension and the superposition dimension is usually computationally inefficient, so we introduce a new notion, named as delta dimension, which totally depends the decomposition matrix, and can be computed without evaluating any high dimensional integral.

We also looked at the error bound of QMC under the effective dimensions based on the formula given by Wang and Fang (2002). If the truncation dimension is small, the first term on the right hand side of (3.14) is much smaller for QMC than for MC. Considering $\| f_u \|$ of the second term on the right hand side of (3.14) is usually small, so all terms are expected to be smaller for QMC than for MC. Thus, if f has low truncation dimension, QMC is usually better than MC. However, when the truncation dimension d_t is large, considering first few leading coordinates of a low discrepancy points, and minimizing $\| f_u \|$ for the rest of dimensions has the potential to improve the QMC results. If the superposition dimension is small (say $d_s \leq 3$), then the superposition discrepancy of low discrepancy points is smaller than random points. Thus, if f has low superposition dimension, QMC is usually better than MC. However when the superposition dimension d_s is larger, minimizing $\| f_u \|$ on the second term of (3.35) is crucial. On the other hand, the effective dimension only provides partial information about the difficulty in approximating integrals. Small effective dimension does not guarantee the effectiveness of QMC. So for higher dimensions, minimizing $\| f_u \|$ will be important.

Due to the importance of effective dimensions, proposing dimension reduction methods becomes the main content of this thesis. In Chapter 4, we review some recently proposed dimension reduction methods, and together with our proposed construction. More clearly, in order to achieve higher convergence rates, alternatives constructions such as PCA and

Brownian bridge are reviewed, but they are equally bad and good for QMC, depending on the function that one wants to integrate. The fast orthogonal transformations proposed by Leobacher (2012) is introduced in this paper. We have seen that for every n by n matrix $\boldsymbol{A}$ with $\boldsymbol{\Sigma} = \boldsymbol{A}\boldsymbol{A}^T$, there exists an orthogonal transform $\boldsymbol{T}$ such that $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{T}$, where $\boldsymbol{L}$ is the Cholesky decomposition of $\boldsymbol{\Sigma}$. However, finding the optimal orthogonal transformation for different pay-off functions is usually hard except that we only take care the first dimension.

The linear transformation considers the structure of the payoff functions. Getting the idea from the structure of payoff, we introduced our new directional control method, which is a more general decomposition method that takes care of the decomposed structure of the payoff functions. We proved that our DC method covers all existing decomposition methods with appropriately choosing our controlling directions and decomposed structures of the payoff function. We also discussed that with specific decomposed structures of the payoff functions, our optimal decompositions could be constructed in different directional control methods: projection control (i.e. column control), sequential control (i.e. row control) and mixture control methods. The orthogonal or projection transformation on f_i's will bring some interesting effects instead of the orthogonal or projection transformation on x_i's due to the reason that transformation on f_i's will not have an internal influence by the function itself. i.e. f_i's contain more information than x_i's in the sense of functional variability, for $i = 1, ..., d$.

Since our DC methods depend on our controlling directions, the structure of the function in its decomposed terms and other criteria such as index function and residual functional variability, the optimal solution for every pay-off function is still under investigation, but the good thing is that our DC method is the only method that controls both rows and columns of the decomposition matrix for all dimensions, and with properly chosen decomposed structure of the payoff function, finding the optimal solution for residual variability (i.e. the remaining directions in which their optimal directions can not be easily determined) with different payoff functions is much easier than other transformation methods, such as orthogonal transformation which can also cover all construction methods, and more flexible than singular value decomposition (SVD), which can only apply to the structure

of $\boldsymbol{x}$. In the case that the effective dimension is not very small and the structure of x_i's are not proportional to the structure of f_i's, for $i = 1, ..., d$, our method will have a significantly strong advantage.

We have seen from the numerical results in Chapter 5 that if the structure of the prices at time t_i is proportional to the structure of the Brownian motions at time t_i for $i = 1, ..., d$, there is not much difference between our DC methods and the others which maximize the explained variability. However, if the structure is not proportional, our method is significantly better than any other method which only consider the explained variability and effective dimensions. If the decomposed structure is in the special case that $f = f_1 + f_2 + ... + f_d$ and the Jacobian matrix of f_i's with respect to x_i's, for $i = 1, ..., d$, is a diagonal matrix, our PC method in the special form that minimize the residual functional variability for all dimensions is numerically equivalent to the LT method. However, the special case that minimizes the effective dimension is not always the best method. We usually need to control the first few dimensions of the PC method by an orthogonal or projection transformation. In the other forms of our Jacobian matrix, our method will be fundamentally different from other methods, and our method is the only method which considers the covariance of different f_i's, i.e. the relationship of f_i's with each other, instead of x_i's. Another interesting result we have seen in our examples is that there always exists a "QMC-friendly" (not definitely the optimal) decomposition constructed by our DC method, the reason is that the decomposition matrix is constructed by its rows and columns, and our method is the only method which considers both directions and their mixture in all dimensions.

Although the research of DC methods is still in progress, we have already seen some values and advantages by applying our methods. There are many different financial applications corresponding to different payoff functions, and our DC methods have some potential to find some other optimized decomposition that may improve the efficiency and accuracy of QMC methods even further. Last but not least, our DC method brings some future researches which can improve the efficiency of simulating financial applications.

6.2 Future Research

Along with our newly designed DC methods, some potential research are under investigation.

1. First (Research on Functional Control), we have seen numerically that with some different decomposed structures of the payoff functions, we could get a high variance reduction. So researching the optimal structure for different pay-off functions will be significantly meaningful with our directional control method. Furthermore, the covariance matrix of the functions f_i's, which is denoted as $\boldsymbol{\Sigma}_f$, does not need to keep full rank. In some situations that Σ has a small rank, our DC method should return better optimal solution than the other methods.

2. Second (Research on Discrepancy Control), we could simulate different f_i's simultaneously with our decomposed structure of the payoff functions by the method of parallel computing.

   For example, suppose we have p independent machines $m_1, ..., m_p$, suppose further that $f = f_1, ..., f_p$, we let machine m_i to evaluate the integration of the function f_i, for $i = 1, ..., p$. Then each m_i will return the evaluation $E(f_i)$ depending on the i^{th} column of p dimensional QMC sequences. If we suppose that $\{Q_1, ..., Q_p\}$ as queues of QMC sequences for the input of machine $\{\ m_1, ..., m_p\ \}$. Then the queue Q_i contains the points of i^{th} column of p dimensional QMC sequences. In this way, the order of evaluation of the points in the queue, will make the convergence rate differently for different machines.

3. Third (Research on Dynamic QMC), when estimating the function f in the additive structure, i.e.

   $$f(\boldsymbol{x}) = f_1(\boldsymbol{x}) + \cdots + f_d(\boldsymbol{x}).$$

   we could estimate the additive term f_d first, and then compute the estimator of $f - E(f_d)$ using the result of $E(f_d)$. The optimal decomposition to estimate $f - E(f_d)$ maybe different because the dimension is lowered. Furthermore, we could estimate

some of the additive terms first, and then compute the estimator of remaining terms using the results that have already been computed and different optimal decompositions. The name "Dynamic"refers to the word "Dynamic Computing", which uses partial estimators already computed to estimate the remaining ones.

4. Fourth (Research on Different Models), every model that we have talked about in this thesis is based on the Black-Scholes assumption, i.e. the Brownian motion, however, our DC method could be used in other models, such as Levy Process, Variance Gamma, etc. The research of applying our DC methods to other financial models may possibly improve the efficiency of QMC even further.

Appendix A

The optimal permutation of Brownian Bridge construction given by Lin and Wang (2008) for d = 2, 4, 8, 16, 32, 64, 128, 256 is given below:

d = 2: 2 1,

d = 4: 3 4 1 2,

d = 8: 6 3 8 1 4 7 2 5,

d = 16: 12 6 15 3 9 16 1 4 7 10 13 2 5 8 11 14,

d = 32: 24 12 30 6 18 3 9 15 21 27 32 1 4 7 10 13 16 19 22 25 28 31 2 5 8 11 14 17 20 23 26 29,

d = 64: 48 24 60 12 36 6 18 30 42 54 63 3 9 15 21 27 33 39 45 51 57 64 1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 2 5 8 11 14 17 20 23 26 29 32 35 38 41 44 47 50 53 56 59 62,

d = 128: 96 48 120 24 72 12 36 60 84 108 126 6 18 30 42 54 66 78 90 102 114 3 9 15 21 27 33 39 45 51 57 63 69 75 81 87 93 99 105 111 117 123 128 1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100 103 106 109 112 115 118 121 124 127 2 5 8 11 14 17 20 23 26 29 32 35 38 41 44 47 50 53 56 59 62 65 68 71 74 77 80 83 86 89 92 95 98 101 104 107 110 113 116 119 122 125,

d = 256: 192 96 240 48 144 24 72 120 168 216 252 12 36 60 84 108 132 156 180 204 228 6 18 30 42 54 66 78 90 102 114 126 138 150 162 174 186 198 210 222 234 246 255 3 9 15 21 27 33 39 45 51 57 63 69 75 81 87 93 99 105 111 117 123 129 135 141 147 153 159 165 171 177 183 189 195 201 207 213 219 225 231 237 243 249 256 1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100 103 106 109 112 115 118 121 124 127 130 133 136 139 142 145 148 151 154 157 160 163 166 169 172 175 178 181 184 187 190 193 196 199 202 205 208 211 214 217 220 223 226 229 232 235 238 241 244 247 250 253 2 5 8 11 14 17 20 23 26 29 32 35 38 41 44 47 50 53 56 59 62 65 68 71 74 77 80 83 86 89 92 95 98 101 104 107 110 113 116 119 122 125 128 131 134 137 140 143 146 149 152 155 158 161 164 167 170 173 176 179 182 185 188 191 194 197 200 203 206 209 212 215 218 221 224 227 230 233 236 239 242 245 248 251 254.

Appendix B

Discrete Cosine Transform:

$$C(x) := \left(\sqrt{\tfrac{2}{n}} \sum_{k=1}^{n} x_k \cos\left(\tfrac{\pi}{n} \left(k - \tfrac{1}{2} \right) \left(j - \tfrac{1}{2} \right) \right) \right)_{j=1}^{n}.$$

Now, we only look at the discrete cosine transform, and we are going to show that LC is approximating to the eigenvalue decomposition, where C is the DCT matrix, and L is defined as

$$L = \frac{1}{\sqrt{n}} \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix}.$$

Proof.:
$(LC)_{lk} = \frac{\sqrt{2}}{n} \sum_{j=1}^{l} \cos\left(\frac{\pi}{n} \left(k - \frac{1}{2} \right) \left(j - \frac{1}{2} \right) \right) = \frac{1}{\sqrt{n(2n)}} \left(\sin\left(\frac{2k-1}{2n} \frac{\pi}{-1} \right) \right)^{-1} \sin\left(\frac{2k-1}{2n} l\pi \right).$
Recall the PCA construction:
$(VD)_{lk} = \lambda_k^{1/2} V_{k,l} = \frac{1}{\sqrt{n(2n+1)}} \left(\sin\left(\frac{2k-1}{2n+1} \frac{\pi}{2} \right) \right)^{-1} \sin\left(\frac{2k-1}{2n+1} l\pi \right).$
We are going to show that for fixed l,k we have $\lim_{n\to\infty} ((LC)_{lk} - (VD)_{lk}) = 0$.
i.e. we found an orthogonal transformation on Cholesky decomposition that is asymptotically equivalent to eigenvalue decomposition.

Now, we reproduce the proof by Leobacher (2010) that the Cosine Decomposition and Eigenvalue decomposition are pretty closed even in a small value of n.
Consider the expected squared Euclidean norm of the difference of two paths generated by two n by n matrices P, Q from the same set of independent standard normals $(X_1, ..., X_n)$:

$$E(\| PX - QX \|) = E\left(\sum_{l=1}^{n}((PX)_l - (QX)_l)^2\right)$$
$$= E\left(\sum_{l=1}^{n}\left(\sum_{k=1}^{n}(P-Q)_{lk}X_k\right)^2\right)$$
$$= \sum_{l=1}^{n}\sum_{k=1}^{n}(P-Q)_{lk}^2 =: d_n(P,Q)^2$$

The last expression is the square of the Euclidean norm of $P - Q$ in R^{n^2}, which is known as the Hilbert-Schmidt norm of $P - Q$.

Theorem .0.1. *For all $n \in N$, we have $d_n(LC, VD) < 1$, and* $\limsup_{n\to\infty} d_n(LC, VD)^2 \leq \frac{2(48-\pi^2)}{(\pi^2-24)^2} \approx 0.381$

Proof: define

$$\delta(n,k) := \sum_{l=1}^{n}(LC - VD)_{lk}^2$$

and

$$u(n,x) := \frac{1}{nx^2}\left(\frac{9\left(\frac{1}{n}\right)+1}{\left(\frac{1}{2n}+1\right)^2\left(6-\left(\frac{1}{2n}+1\right)^2x^2\right)^2} + \frac{9}{(6-x^2)^2} - \frac{\sqrt{n}\sqrt{4n+2}}{2n+1}\right).$$

Lemma .0.1. *u is increasing in the second variable x on $\left[0, \frac{\pi}{2}\right]$.*

Proof: Compute the partial derivative of u w.r.t. x as $\frac{\partial}{\partial x}u(n,x)$, then it will be easy to check that $\frac{\partial}{\partial x}u(n,x) > 0$ on $\left[0, \frac{\pi}{2}\right]$. We leave the details to the reader.
i.e. u is increasing in the second variable x on $\left[0, \frac{\pi}{2}\right]$. □

Lemma .0.2. *For all n and all $1 \leq k \leq n$, we have*

$$\delta(n,k) < u\left(n, \frac{2k-1}{2n+1}\frac{\pi}{2}\right).$$

Proof: The sum $\delta(n,k) := \sum_{l=1}^{n}(LC-VD)_{lk}^2$ can be computed by writing $sin(x) = \frac{1}{2i}\left(e^{ix}-e^{-ix}\right)$ for $x \in \{\frac{2k-1}{2n}l\pi, \frac{2k-1}{2n+1}l\pi\}$ so that the sum becomes a sum of 4 geometric sums.

Using the some facts that $\sin(\pi(2k-1)) = 0$, $\sin\left(\frac{\pi(2k-1)(2n+1)}{2n}\right) = -\sin\left(\frac{\pi(2k-1)}{2n}\right)$ and $\sin\left(\frac{\pi(2k-1)(4n+1)}{4n}\right) = -\sin\left(\frac{\pi(2k-1)}{4n}\right)$, for integers k, n. The result can be simplified to

$$\delta(n,k) = \frac{1}{4n}\left(\frac{1}{n}+1\right)\csc^2\left(\frac{\pi(2k-1)}{4n}\right) + \frac{1}{4n}\csc^2\left(\frac{\pi(2k-1)}{2(2n+1)}\right)$$
$$-\frac{\sqrt{2}\csc\left(\frac{\pi(2k-1)}{2(2n+1)}\right)}{4n\sqrt{n(2n+1)}}\left(\csc\left(\frac{\pi(2k-1)}{4n(2n+1)}\right)+\csc\left(\frac{\pi(2k-1)(4n+1)}{4n(2n+1)}\right)\right).$$

Next we use that for $x \in \left(0, \frac{\pi}{2}\right)$, one has $x - \frac{x^3}{6} < \sin(x) < x$ and, therefore,

$$\frac{1}{x-\frac{x^3}{6}} > \csc(x) > \frac{1}{x}.$$

After some simplifications, we get the estimate

$$\delta(n,k) < \frac{1}{n\left(\frac{\pi(2k-1)}{2(2n+1)}\right)^2}\left(\frac{9\left(\frac{1}{n}+1\right)}{\left(\frac{1}{2n}+1\right)^2\left(6-\left(\frac{1}{2n}+1\right)^2\left(\frac{\pi(2k-1)}{2(2n+1)}\right)^2\right)^2} + \frac{9}{\left(6-(\frac{\pi(2k-1)}{2(2n+1)})^2\right)^2} - \frac{\sqrt{n}\sqrt{4n+2}}{4n+1}\right).$$

Therefore, $\delta(n,k) < u(n, \frac{2k-1}{2n+1}\frac{\pi}{2})$. □

Lemma .0.3. *The sequence $d_n(LC,VD)^2$ is bounded by $\frac{2(48-\pi^2)}{(\pi^2-24)^2}$.*

Proof: by Lemma .0.1 and .0.2, we have

$$d_n(LC,VD)^2 = \sum_{l=1}^{n}\sum_{k=1}^{n}(LC-VD)_{lk}^2 = \sum_{k=1}^{n}\delta(n,k) < nu\left(n, \tfrac{\pi}{2}\right).$$

And it is easy to check that

$\lim_{n\to\infty} nu\left(n, \frac{\pi}{2}\right) = \frac{2\left(48-\pi^2\right)}{\left(\pi^2-24\right)^2} \approx 0.381.$ □

Hence, by Lemma .0.3 and definition of $u(n,x)$, we can get

$$\begin{aligned} d_n(SC,VD)^2 &< nu\left(n,\frac{\pi}{2}\right) \\ &\leq \frac{4}{\pi^2}\left(\frac{2}{\left(\frac{1}{2n}+1\right)^2\left(2-(\frac{1}{2n}+1)^2\frac{\pi^2}{12}\right)^2-(\frac{1}{2}-\frac{1}{64n^2})}\right). \end{aligned}$$

Now use the estimates $\frac{4}{\pi^2} < 0.41$ and $\frac{\pi^2}{12} < 0.83$ to verify that $nu\left(n, \frac{\pi}{2}\right) < 1$ for all $n \geq 3$. For $n = 1, 2$, we can use direct computation to show that $d_n(SC, VD) < 1$. □

Hence, we can conclude that for fixed l,k we have $\lim_{n\to\infty}((LC)_{lk} - (VD)_{lk}) = 0$. □

Appendix C

Here's the proof of Theorem 4.1.4 according to Wang and Tan (2012).

Proof: From (4.9), we have

$$\boldsymbol{q}^T \boldsymbol{A_0} = D\boldsymbol{U_1}^T.$$

Define the columns of $\boldsymbol{U}$ as $\boldsymbol{U_1}, ..., \boldsymbol{U_d}$. Under the transformation $\boldsymbol{x} = \boldsymbol{A_0 U z}$, we have

$$\boldsymbol{q}^T \boldsymbol{x} = \boldsymbol{q}^T \boldsymbol{A_0 U z} = D\boldsymbol{U_1}^T(\boldsymbol{U_1} z_1 + \boldsymbol{U_2} z_2 + \cdots + \boldsymbol{U_d} z_d) = D z_1,$$

by the orthogonality of the columns $\boldsymbol{U_1}, ..., \boldsymbol{U_d}$ of $\boldsymbol{U}$. Therefore,

$$h(\boldsymbol{q}^T \boldsymbol{x}) = h(D z_1).$$

This proves the first part of the theorem.

The remaining results follow form the following equivalences (under the transformations $\boldsymbol{x} = \boldsymbol{A_0 U z}$ and $\boldsymbol{z} = \Phi^{-1}(\boldsymbol{u})$) that

$$\{h(\boldsymbol{q}^T \boldsymbol{x} < H\} \Longleftrightarrow \{h(D z_1) < H\} \Longleftrightarrow \{u_1 < c\}. \square$$

References

[1] Acworth, P., Broadie, M. and Glasserman, P. (1998). A comparison of some Monte Carlo and quasi-Monte Carlo techniques for option pricing. In Monte Carlo and Quasi-Monte Carlo Methods 1996 (H. Niederreiter, P. hellekalek, G. Larcher & P. Zinterhof eds). New York: Springer, 1-18.

[2] Akesson, F. and Lehoczky, J.P. (1998). Discrete eigenfunction expansion of the multi-dimensional Brownian motion and the ornstein-Uhlenbeck process. *Technical Report*, Carnegie-Mellon University.

[3] Avramidis, A.N. and L'Ecuyer, P. (2006). Efficient Monte Carlo and quasi-Monte Carlo option pricing under the variance gamma model. *Management Science*, **52**(12): 1930-1944.

[4] Barth, T.J., Griebel, M., Keyes, D.R., Nieminen, R.M., Roose,D. and Schlick, T. (2011). Lecture Notes in Computational Science and Engineering. *Mathematics Subject Classification (2010): 65-02*, Springer-Verlag Berlin Heidelberg.

[5] Benth, F.E., Groth, M. and Kettler, P.C. (2006). A quasi-Monte Carlo algorithm for the normal inverse Gaussian distribution and valuation of financial derivatives. *International Journal of Theoretical and Applied Finance*, **9**(6): 843-867.

[6] Boyle, P. (1977). Options: A Monte Carlo Approach. *J. Financial Economics*, **4**(3): 323-338.

[7] Boyle, P., Broadie, M. and Glasserman, P. (1997). Monte Carlo methods for security pricing. *J. Econom. Dynam. Control*, **21**: 1267-1321.

[8] Boyle, P., Lai, Y., and Tan, K.S. (2005). Pricing options using lattice rules. *North Amer. Act. J.*, 50-76.

[9] Braaten, E. and Weller, G. (1979). An improved low-discrepancy sequence for multidimensional quasi-Monte Carlo integration. *J. Computational Physics*, **33**: 249-258.

[10] Bracewell, R.N. (2000). The Fourier Transform and its Applications. *McGrawHill, third edition.*

[11] Bratley, P., Fox, B.L. and Niederreiter, H. (1992). Implementation and tests of low-discrepancy sequences. *ACM Transactions on Modeling and COmputer Simulation*, **2**: 195-213.

[12] Caflisch, R., Morokoff, W. and Owen, A. (1997). Valuation of mortgage backed securities using Brownian bridges to reduce effective dimension. *J. Comp. Finance*, **1**: 27-46.

[13] Detemple, J. and Rindisbacher, M. (2011). Monte Carlo methods for derivatives of options with discontinuous payoffs. *U.U.D.M. Project Report*, Department of Mathematics, Wake Forest University.

[14] Drmota, M. and Tichy, R.F. (1997). Sequences, discrepancies and applications. *Lecture Notes in Mathematics*, **1651**, Springer-Verlag, Berlin.

[15] Fang, K.T., Hickernell, F.J. and Niederreiter, H. (2000). Monte Carlo and quasi-Monte Carlo methods 2000. *Hong Kong*, Springer.

[16] Faure, H. (1982). Discrépance de suites associées *à* un système de numération (en dimensions). *Acta Arith.*, **41**: 337-351.

[17] Glasserman, P. (2004). Monte Carlo methods in financial engineering. New York, Springer-Verlag.

[18] Griebel, M. and Holtz M. (2010). Dimension-wise integration of high-dimensional functions with applications to finance. *J. Complexity*, **26**(5): 455-489.

[19] Halton, J.H. (1960). On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik*, **2**: 84-90.

[20] Halton, J.H. (1964). Algorithm 247: Radical-inverse quasi-random point sequence. *ACM*, 701.

[21] Hardy, G.H. (1905). On double Fourier series, and especially those which represent the double zeta-function with real and incommensurable parameters. *Quarterly Journal of Mathematics*, **37**: 53-79.

[22] Hickernell, F.J. (1996). The mean square discrepancy of randomized nets. *ACM Trans. Model. Comput. Simul.*, **6**: 274-296.

[23] Hickernell, F.J. (1998). A generalized discrepancy and quadrature error bound. *J. Mathematics of Computation*, **67**(221): 299-322.

[24] Imai, J. and Tan, K. S. (2006). A general dimension reduction technique for derivative pricing. *J. Computational Finance*, **10**(2): 129-155.

[25] Jiang, M. and Mcnamara, E. (2002). Evaluating the quasi-Monte Carlo method for discontinuous integrands. *Technical Report*, 1-17.

[26] Joy, C., Boyle, P. and Tang, K. S. (1996). Quasi-Monte Carlo methods in numerical finance. *Management Science*, **42**(6): 926-938.

[27] Keiner, J. and Waterhouse, B.J. (2009). Fast principal components analysis method for finance problems with unequal time steps. *Monte Carlo and Quasi-Monte Carlo Methods 2008*, Part 3, 455-465.

[28] Krause, M. (1903a). *Ü*ber mittelwerts*ätze* im gebiete der doppelsummen and doppelintegrale. *Leipziger Ber.*, **55**: 239-263.

[29] Krause, M. (1903b). *Ü*ber Fouriersche reihen mit zwei veränderlichen grössen. *Leipziger Ber.*, **55**: 164-197.

[30] L'Ecuyer P. and Lemieux, C. (2002). Recent advances in randomized quasi-Monte Carlo methods. *Modeling Uncertainty: An Examination of Stochastic Theory, Methods,*

and Applications, M. Dror, P. L'Ecuyer, and F. Szidarovszki, eds., Kluwer Academic,, 419-474.

[31] Lemieux, C. (2009). Monte Carlo and Quasi-Monte Carlo Sampling. *Springer Series in Statistics*, New York, 2009.

[32] Lemieux, C. and L'Ecuyer P. (2001). On the use of quasi-Monte Carlo methods in computational finance. *Lecture Notes in Computer Science*, **2073**.

[33] Larcher, G., Leobacher, G. and Scheicher, K. (2003). On the tractability of the Brownian bridge algorithm. *J. Complexity*, **19**: 511-528.

[34] Leobacher, G. (2012). Fast orthogonal transforms and generation of Brownian paths. *J. Complexity*, **28**: 278-302.

[35] Lin, J., and Wang, X. (2008). New Brownian bridge construction in quasi-Monte Carlo methods for computational finance. *J. Complexity*, **24**: 109-133.

[36] Matousek, J. (1998). On the L_2-discrepancy for anchored boxes. *J. Complexity*, **14**: 527-556.

[37] Morokoff, W.J. and Caflisch, R.E (1994). Quasi-random sequences and their discrepancies. *S.I.A.M Journal Scienti"c Computing*, **15**: 1251-1279.

[38] McKay, M. D., Conover, W. J. and Beckman, R. J. (1979). A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics*, **21**: 239-245.

[39] Moore, E.H. (1920). On the Reciprocal of the General Algebraic Matrix. *Bulletin ofthe American Mathematical Society*, **26**: 394-395.

[40] Moskowitz, B. and Caflisch, R. E. (1996). Smoothness and dimension reduction in quasi-Monte Carlo methods. *J. Mathematical and Computer Modeling*, **23**: 37-54.

[41] Niederreiter, H. (1992). Random number generation and quasi-Monte Carlo methods. *SIAM*, Philadelphia.

[42] Oehlert, Gary W. (1992). A Note on the Delta Method. *The American Statistician*, **46**(1): 27-29.

[43] Okten, G. (2009). Generalized von Neumann-Kakutani transformation and random-start scrambled Halton sequences. *J. Complexity*, **25**: 318-331.

[44] Owen, A.B. (1995). Randomly permuted (t,m,s)-nets and (t,s)-sequences. In H. Niederreiter and P. J.-S. Shiue Eds., Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing. *Lecture Notes in Statistics*, **106**: 299-317. Springer-Verlag, New York.

[45] Owen, A.B. (2001). The dimension distribution, and quadrature test functions. *Technical Report, Stanford University.*

[46] Palan, S. (2010). Digital options and efficiency in experimental asset markets. *J. Economic Behavior & Organization*, **75**(3): 506-522.

[47] Papageorgiou, A. (2001). Fast convergence of quasi-Monte Carlo for a class of isotropic integrals. *Math. Comp.*, **70**: 297-306.

[48] Papageorgiou, A. (2002). The Brownian bridge does not offer a consistent advantage in quasi-Monte Carlo integration. *J. Complexity*, **18**: 171-186.

[49] Paskov, S. H. and Traub, J. F. (1995). Faster evaluation of financial derivatives. *J. Portfolio Management*, **22**(1): 113-120.

[50] Petroni, N.C. and Sabino, P. (2011). Pricing and Hedging Asian Basket Options with Quasi-Monte Carlo Simulations. *Methodology and Computing in Applied Probability*, 1-17.

[51] Roth, K.F. (1954). On irregularities of distribution. *Mathematika*, **1**: 73-79.

[52] Scheicher, K. (2007). Complexity and effective dimension of discrete Lévy areas. *J. Complexity*, **23**(2): 152-168.

[53] Wang, X. and Sloan, I.H. (2010). Quasi-Monte Carlo methods in financial engineering: an equivalent principle and dimension reduction. *Oper. Res.*, to appear.

[54] Sloan, I.H. and Wozniakowski, H. (1998). When are quasi-Monte Carlo algorithms efficient for high dimensional integrals? *J. Complexity*, **14**: 1-33.

[55] Sloan, I.H. and Wozniakowski, H. (2001). Tractability of multivariate integration for weighted Korobov classes. *J. Complexity*, **17**: 697-721.

[56] Sobol', I.M. (1967). On the distribution of points in a cube and the approximate evaluation of integrals. *Zh. Vychisli. Mat. i Mat. Fiz.*, **7**: 784-802.

[57] Sobol', I.M. and Kucherenko, S.S. (2005). Global Sensitivity Indices for Nonlinear Mathematical Models. *Review, Wilmott*, **1**: 56-61.

[58] Tuffin, B. (1997). Simulation accélérée par les méthodes de Monte Carlo et Quasi-Monte Carlo: théorie et applications. Thèse de Doctorat. *Université de Rennes*, **1**.

[59] Tuffin, B. (1997). Variance Reductions applied to product-form multi-class queuing network. *ACM Transactions on Modeling and Computer Simulation*, **7**(4): 478-500.

[60] Tuffin, B. (2008). Randomization of quasi-Monte Carlo methods for error estimation: survey and normal approximation. *Monte Carlo Methods and Applications mcma*, **10**(3-4): 617-628.

[61] van der Corput, J.G. (1935). Verteilungsfunktionen. *Proc. ned. Akad. v. Wet.*, **38**: 813-821.

[62] Wang, Z. (1984). Fast algorithms for the discrete W transform and for the discrete Fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **32**(4).

[63] Wang, X. and Fang, K.T. (2003) The effective dimension and quasi-Monte Carlo integration. *J. Complexity*, **19**: 101-124.

[64] Wang, X. and Sloan, I.H. (2008). Low discrepancy sequences in high dimensions: how well are their projections distributed. *J. Computational and Applied Mathematics*, **213**(2): 366-386.

[65] Wang, X., and Tan, K.S. (2012). How do path generation methods affect the accuracy of quasi-Monte Carlo methods for problems in finance? *J. Complexity*, **28**(2): 250-277.

[66] Wang, X., and Tan, K.S. (2012). Pricing and hedging with discontinuous functions: quasi-Monte Carlo methods and dimension reduction. *Management Science*, in progress.

[67] Zhang, H. (2009). Pricing Asian options using Monte Carlo methods. *Computational Statistics & Data Analysis*, **51**(7): 3393-3417.